

PATENT ABSTRACTS OF JAPAN

REFERENCE 4

(11)Publication number : 07-262059

(43)Date of publication of application : 13.10.1995

(51)Int.Cl.

G06F 12/00

G06F 12/00

G11B 27/00

(21)Application number : 06-055376

(71)Applicant : SONY CORP

(22)Date of filing : 25.03.1994

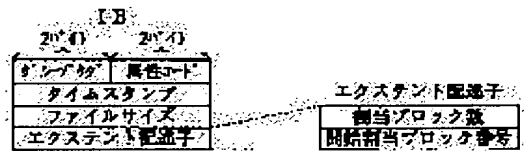
(72)Inventor : SUZUKI YUICHI

(54) FILE MANAGEMENT METHOD

(57)Abstract:

PURPOSE: To enable even such an equipment that has no character input means to perform a fast access to a file with a small memory capacity.

CONSTITUTION: The cluster of a data recording medium consists of 32 sectors and a single allotment block consists of four sectors. A file is recorded with an allotment block defined as minimum unit. The data recording medium also includes a bit map sector and an index sector and controls the using state of each allotment block. The index blocks are recorded on the index sector, and group tag attribute code, time stamp, file size and extent describer (the allotment block number and the start allotment block number) are recorded on each index block.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-262059

(43) 公開日 平成7年(1995)10月13日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 0 1 H	7608-5B		
	5 2 0 J	7608-5B		
G 1 1 B 27/00		D 8224-5D		
		8224-5D	G 1 1 B 27/ 00	D

審査請求 未請求 請求項の数17 O L (全 28 頁)

(21) 出願番号 特願平6-55376

(22) 出願日 平成6年(1994)3月25日

(71) 出願人 000002185

ソニー株式会社

東京都品川区北品川6丁目7番35号

(72) 発明者 鈴木 雄一

東京都品川区北品川6丁目7番35号 ソニー株式会社内

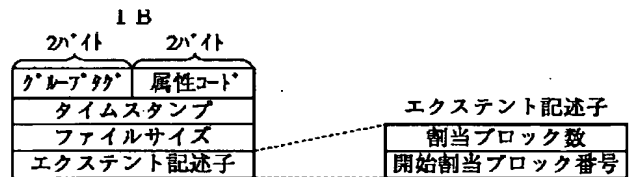
(74) 代理人 弁理士 稲本 義雄

(54) 【発明の名称】 ファイル管理方法

(57) 【要約】

【目的】 文字入力手段を持たない機器においても、小さなメモリ容量で、高速なファイルアクセスを可能にする。

【構成】 データ記録媒体のクラスタを32セクタにより構成し、4セクタにより1つの割当ブロックを構成する。ファイルを割当ブロックを最小単位として記録するようにする。また、記録媒体には、ビットマップセクタとインデックスセクタを設け、ビットマップセクタにより、割当ブロックの使用状態を管理する。インデックスセクタには、インデックスブロックを記録し、各インデックスブロックには、グループタグ、属性コード、タイムスタンプ、ファイルサイズ、エクステント記述子（割当ブロック数および開始割当ブロック番号）を記録する。



I B の構成列

【特許請求の範囲】

【請求項 1】 多数のブロックに区分した領域にファイルを記録し、再生するようにした記録媒体のファイルを管理するファイル管理方法において、

前記記録媒体に、前記ブロックの使用状態を表す使用情報を記録するブロック管理領域を形成するとともに、前記ファイルの記録に使用された前記ブロックに関するインデックス情報を記録するインデックス領域とを形成し、

前記インデックス領域に、前記ファイルを関連づける関連情報を記録することを特徴とするファイル管理方法。

【請求項 2】 前記関連情報は、前記ファイルをグループに分類して管理するグループタグを含むことを特徴とする請求項 1 に記載のファイル管理方法。

【請求項 3】 前記関連情報は、親の前記ファイルを特定する番号を記録するインデックスタグを含むことを特徴とする請求項 1 に記載のファイル管理方法。

【請求項 4】 前記関連情報は、連続する前記インデックス領域を特定する情報を記録するリンクタグを含むことを特徴とする請求項 1 または 3 に記載のファイル管理方法。

【請求項 5】 前記関連情報は、前記リンクタグに連続する前記インデックス領域を特定する情報が記録されているか否かを表すリンクフラグをさらに含むことを特徴とする請求項 4 に記載のファイル管理方法。

【請求項 6】 前記関連情報は、1 つの前記ファイルを管理する複数の前記インデックス領域のうち、先頭のインデックス領域であるか否かを表す拡張フラグをさらに含むことを特徴とする請求項 1、3 または 4 に記載のファイル管理方法。

【請求項 7】 複数の前記インデックス領域に記録された情報によりインデックスページを形成し、前記インデックスページの 1 ページ分の情報を記憶することができるページバッファメモリを複数ページ分設け、

前記ページバッファメモリを管理する情報を記憶するページタグを形成し、

前記ページタグに、前記インデックスページを特定する番号、前記ページバッファメモリを指定するポインタ、前記ページバッファメモリの内容が更新されたか否かを表す更新情報の少なくとも 1 つを記録することを特徴とする請求項 1、4 または 5 に記載のファイル管理方法。

【請求項 8】 前記記録媒体に、前記インデックスページにおける未使用インデックス領域の数を表すページアカウントを記録するページアカウント領域を形成することを特徴とする請求項 7 に記載のファイル管理方法。

【請求項 9】 前記記録媒体のページアカウント領域より読み出した前記ページアカウントを記憶するページアカウントテーブルをメモリに形成することを特徴とする請求項 8 に記載のファイル管理方法。

【請求項 10】 前記インデックス領域には、グループタグ、インデックスタグ、属性コード、タイムスタンプ、ファイルサイズおよびエクステント記述子の少なくとも 1 つを有するインデックスブロックを記録することを特徴とする請求項 1 に記載のファイル管理方法。

【請求項 11】 前記インデックスブロックは、リンクされた一連の前記インデックスブロックの先頭の前記インデックスブロックに適用される基本型フォーマットと、他の前記インデックスブロックに続く前記インデックスブロックに適用される拡張型フォーマットとにより構成されていることを特徴とする請求項 10 に記載のファイル管理方法。

【請求項 12】 複数の前記インデックスブロックによりインデックスページを形成し、前記記録媒体に、前記インデックスページにおける未使用の前記インデックスブロックの数を表すページアカウントを記録するページアカウント領域を形成することを特徴とする請求項 11 に記載のファイル管理方法。

【請求項 13】 前記基本型フォーマットのインデックスブロックは、前記ページアカウントから検索した 2 以上の未使用インデックスブロックが存在する前記インデックスページに記録することを特徴とする請求項 12 に記載のファイル管理方法。

【請求項 14】 前記拡張型フォーマットのインデックスブロックを記録する前記インデックスページは、1 つ前の前記拡張型フォーマットのインデックスブロックまたは基本型フォーマットの前記インデックスブロックが属する前記インデックスページの前記ページアカウントから昇順に検索することを特徴とする請求項 12 または 13 に記載のファイル管理方法。

【請求項 15】 前記インデックスブロックには、前記属性コードとして、ディレクトリファイルであるか否かを表すコードを記録し、

前記ディレクトリファイルのデータとして子ファイルのインデックス番号を記録し、

前記子ファイルの前記インデックスブロックには、前記インデックスタグとして、前記ディレクトリファイルのインデックス番号を記録することを特徴とする請求項 10 に記載のファイル管理方法。

【請求項 16】 前記ディレクトリファイルのデータとして、前記子ファイルのインデックス番号とともに、前記子ファイルの名称を記録することを特徴とする請求項 15 に記載のファイル管理方法。

【請求項 17】 前記属性コードは、複数の種類の前記ディレクトリファイルのいずれかを表す複数ビットのコードであることを特徴とする請求項 15 または 16 に記載のファイル管理方法。

【発明の詳細な説明】**【0001】**

【産業上の利用分野】 本発明は、例えば、デジタルスチ

ルカメラ等において、記録したデータ（ファイル）を管理する場合に用いて好適なファイル管理方法に関するものである。

【0002】

【従来の技術】図32は、従来のファイル管理方法を適用したデータ記録再生装置の構成例を示す。データ記録媒体1は、例えば磁気ディスク、光磁気ディスク、ICメモ리카ード等により構成される。媒体駆動部2は、例えば上記光磁気ディスクに対しては、スピンドルモータ、光ピックアップ、サーボ回路、変復調回路等より構成されるディスクドライブ装置である。

【0003】メモリ3は、データの記録、再生時において、一時的にデータを記憶するバッファや、各種管理情報等のワークエリアとして使用される。CPU4は、内蔵するプログラムに従って、媒体駆動部2の制御、メモリ3上の管理情報の操作、データ転送等を実行する。

【0004】データ入力部5は、例えばデジタルインターフェース、キーボード、A/Dコンバータ等により構成される。データ出力部6は、例えばデジタルインターフェース、CRTディスプレイ、D/Aコンバータ等により構成される。

【0005】データの記録時においては、データ入力部5より入力されたデータは、CPU4の働きにより、一旦メモリ3の所定の領域に転送される。次に、CPU4の制御により、媒体駆動部2が動作し、メモリ3上のデータを、データ記録媒体1に記録する。

【0006】データの再生時においては、CPU4の制御により、媒体駆動部2が動作し、データ記録媒体1に記録されているデータが、メモリ3の所定の領域に一旦転送される。次に、CPU4の働きにより、メモリ3上のデータを読み出し、データ出力部6により、出力装置（図示せず）に出力する。

【0007】このデータ記録再生装置は、データ記録媒体1に対し、例えば音楽データであれば複数の曲、静止画データであれば複数枚の画像など、複数のデータを自由に記録し、再生し、消去するものである。この機能を実現するため、このデータ記録再生装置上に、ファイルシステムという機構を設けるのが一般的である。

【0008】ファイルシステムの代表的な役割は、以下のとおりである。

1. データ記録媒体1の空き領域を管理し、ファイルの記録に際して、重ね書きによるデータの喪失を防止する。
2. データ記録媒体1の記録済領域を管理し、ファイルの再生に際して、そのファイルのデータの記録場所を知る。
3. ファイル識別子を管理し、ファイル識別子によるファイル（データ）へのアクセスを実現する。

【0009】このようなデータ記録再生装置のファイルシステムの代表例として、コンピュータシステムにおけ

るディスク装置上のファイルシステムがある。

【0010】図33は、MS-DOS（商標）のファイルシステムの構成を略化して表している。MS-DOSにおけるデータ記録媒体1は、ファイルアロケーションテーブル（以下、FATと表す）領域11、ルートディレクトリ領域12、およびデータ領域13で構成されている。

【0011】データ領域13は、1024バイトよりなる、2乃至N+1のN個のクラスタにより構成されている。例えば、データ領域13が1M（メガ）バイトのデータ記録媒体1は、クラスタ2乃至1025クラスタの1024個のクラスタで構成されている。

【0012】FAT領域11にはFATが形成され、このFATは、2乃至N+1のN個のFATエントリで構成されており、1個のFATエントリ（12ビット）は、1個のクラスタに対している。以下に、FATの内容を示す。

【0013】（1）FATエントリの000は、そのクラスタが使用可能であることを示す。

（2）FATエントリの002乃至FF6は、そのクラスタにはファイル（データ）が記録されており、そのファイルは、その番号が示す次のクラスタに続いていることを表す。

（3）FATエントリのFF8乃至FFFは、そのクラスタにはファイル（データ）が記録されており、それがファイルの最終クラスタであることを示す。

（4）FATエントリのFF7は、そのクラスタが不良であることを示す。

【0014】上記ルールに従えば、図33の例においては、クラスタ2、4、6よりなる1個のファイルと、クラスタ3よりなる1個のファイルが、データ記録媒体1に記録されており、クラスタ5およびクラスタ7乃至N+1は、空きとなっている。

【0015】図34は、ルートディレクトリ領域12に形成されているルートディレクトリの構成図である。ルートディレクトリは、それぞれ32バイトよりなる、0乃至M-1のM個のディレクトリエントリで構成されており、典型的なMの値は112である。各ディレクトリエントリには、ファイル名、ファイル属性、更新日時、開始クラスタ、ファイルサイズが記録されている。

【0016】ファイル名の第1バイトの00またはE5は、そのディレクトリエントリが空いていることを表す。このディレクトリエントリには、ファイル属性として、ディレクトリ属性をなす、特別なファイルを登録することができ、これをディレクトリファイルと呼ぶ（このファイル（データ）自体は、データ領域13に記録される）。

【0017】図35は、ディレクトリファイルによる、階層的なファイル管理の例を表している。ルートディレクトリにおいては、第1のエントリに、“FILE1”

のファイル名と、そのデータがクラスタ2（データ領域13）に記録されていることが登録されている。第2のエントリには、“FILE2”のファイル名と、そのデータがクラスタ3（データ領域13）に記録されていることが登録されている。第3のエントリには、“DIR1”のファイル名と、そのデータがクラスタ4（データ領域13）に記録されていることが登録されている。また、ファイル属性として、ディレクトリ属性が記録されており、クラスタ4の内容が、ディレクトリファイル（サブディレクトリ）であることが示されている。

【0018】クラスタ4（データ領域13）のディレクトリファイルにおいては、第1のエントリに、“FILE3”のファイル名と、そのデータがクラスタ5（データ領域13）に記録されていることが登録されている。第2のエントリには、“FILE4”のファイル名と、そのデータがクラスタ6（データ領域13）に記録されていることが登録されている。

【0019】上記構成によるファイルシステムにデータを記録するときは、図36のフローチャートに示す手順で行う。

【0020】ステップS301で、FATを走査し、値が000（使用可能）のFATエントリを、必要個数見つけ出す。次にステップS302で、見つけたFATエントリに対応する各クラスタに、データを記録する。さらにステップS303では、データを記録したクラスタに対応する各FATエントリに、次のクラスタ番号またはFFFを記録し、データが記録されていることを示す。

【0021】次にステップS304で、ルートディレクトリ（またはディレクトリファイル）を走査し、空いているディレクトリエントリを見つけて出す。そしてステップS305で、見つけたそのディレクトリエントリに、ファイル名、ファイル属性、更新日時とともに、上記ステップS302で記録した最初のクラスタ番号を記録し、さらにファイルサイズを記録する。

【0022】上記構成によるファイルシステムからデータを再生するときは、図37のフローチャートに示す手順で行う。

【0023】最初に、ステップS321で、ルートディレクトリ（またはディレクトリファイル）を走査し、目的の（再生対象とされる）ファイル名と一致するディレクトリエントリをみつける。次にステップS322で、見つけたディレクトリエントリに記録されている開始クラスタ番号（図34）を読み、そのクラスタからデータを読み出す。ステップS323では、いまデータを読み出したクラスタに対応するFATエントリを読み、ステップS324で、その値を判定する。その値が002乃至FF6（次のクラスタの番号）ならば、ステップS325で、次のクラスタからデータを読み出し、ステップS323に戻り、同様の処理を繰り返す。

【0024】FATエントリの値が、002乃至FF6でないとき、ステップS326で、FF8乃至FFF（最終クラスタ）か否かを判定し、そうならば、終了する。そうでない場合、ステップS327で、エラー処理を行い、終了する。

【0025】上記構成によるファイルシステムからデータを消去するときは、図38のフローチャートに示す手順で行う。

【0026】最初に、ステップS341で、ルートディレクトリ（またはディレクトリファイル）を走査し、目的の（消去対象とされる）ファイル名と一致するディレクトリエントリをみつける。次にステップS342で、見つけたディレクトリエントリに記録されている開始クラスタ番号を読み、対応するFATエントリの値を得る。

【0027】ステップS343では、そのFATエントリに000（使用可能）を記録し、そのクラスタを使用可能とする。また、ステップS344では、ステップS342で読み取ったFATエントリの値が002乃至FF6（次のクラスタ有）か否かを判定し、次のクラスタが存在するのであれば、ステップS345に進み、次のFATエントリの値を得て、ステップS343に戻り、それ以降の処理を繰り返す。

【0028】FATエントリの値が、002乃至FF6ではないとき（次のクラスタが存在しないとき）、ステップS346に進み、FF8乃至FFF（最終クラスタ）であるか否かを判定し、そうであったら、ステップS347へ移行する。ステップS347では、ステップS341で見つけたディレクトリエントリにおいて、ファイル名の第1バイトにE5（空き）を記録して、そのエントリを消去する。FATエントリの値が、002乃至FF6でもないときは、ステップS348に進み、エラー処理を実行する。

【0029】次に、コンピュータのOSのもう1つの代表例であるUNIX（商標）のファイルシステムについて説明する。UNIXのファイルシステムは非常に複雑なので、簡略化したモデルにより説明する。

【0030】図39は、UNIXのファイルシステムの構成を表している。UNIXにおいては、データ記録媒体1に、ビットマップ領域21、iノードリスト領域22、およびデータ領域23が形成されている。

【0031】データ領域23は、それぞれ1024バイトよりなる、0乃至N-1のN個の論理ブロックで構成されている。例えば、データ領域23が1Mバイトのとき、0乃至1023の1024個の論理ブロックで構成されている。

【0032】ビットマップ領域21に形成されるビットマップは、各論理ブロックにつき各1ビットよりなる、Nビットのフラグにより構成されており、以下の内容を示す。

【0033】(1) フラグの1は、その論理ブロックが使用可能であることを表す。

(2) フラグの0は、その論理ブロックにデータが記録されているか、不良ブロックであることを表す。

【0034】図40は、iノードリスト領域22に形成されるiノードリストの構成を表している。iノードリストは、2乃至M+1のM個のiノードで構成される。このうち、iノード2は、ルートディレクトリのために確保されている。上記iノードは、ファイルのモード(属性など)、タイムスタンプ、ファイルサイズ、参照フラグ、直接ポインタ、間接ポインタにより構成されている。

【0035】参照フラグが1のとき、そのiノードが使用されていることを示し、0のとき、空きであることを示す。

【0036】直接ポインタには、データが記録された論理ブロック番号(0乃至N-1)が記録される。図40の例では、0が記録されており、論理ブロック0(直接ブロック)に、ファイルをなす第1のデータが記録されていることを示している。間接ポインタには、間接ブロック番号が記録され、間接ブロックには、データが記録された論理ブロック番号が記録される。図40の例では、1が記録されており、論理ブロック1が間接ブロックであることを示している。

【0037】図40において、間接ブロック(論理ブロック1)には、3個の直接ポインタが記録されており、第1の直接ポインタは、論理ブロック2にファイルをなす第2のデータが記録されていることを示し、第2の直接ポインタは、論理ブロック3にファイルをなす第3のデータが記録されていることを示し、第3の直接ポインタは、論理ブロック4にファイルをなす第4のデータが記録されていることを示している。

【0038】UNIXにおいては、ディレクトリもファイルの一種(ディレクトリファイル)とされ、データ領域23に記録される。図41は、このディレクトリファイルの構成を表している。ディレクトリファイルは、複数のディレクトリエントリより構成されており、各ディレクトリエントリは、エントリの大きさ、ファイル名の長さ、iノード番号、ファイル名より構成されている。

【0039】この例では、第1のエントリに、ファイル"FILE1"が登録されており、そのファイルの諸情報は、iノード3に記録されていることが示されている。第2のエントリには、ファイル"FILE2"が登録されており、そのファイルの諸情報は、iノード4に記録されていることが示されている。第3のエントリには、ファイル"DIR1"が登録されており、そのファイルの諸情報は、iノード5に記録されていることが示されている。

【0040】これを受けて、iノード3には、直接ポインタにより、FILE1のデータが論理ブロック5に記

録されていることが示されている。また、iノード4には、直接ポインタにより、FILE2のデータが論理ブロック6に記録されていることが示されている。さらに、iノード5には、直接ポインタにより、DIR1のデータが論理ブロック7に記録されていることが示されている。また、iノード5のモードは、ファイル"DIR1"がディレクトリファイルであることを示している。

【0041】論理ブロック7には、DIR1の内容(サブディレクトリ)が記録されており、この例では、2つのエントリが登録されている。第1のエントリには、ファイル"FILE3"が登録されており、そのファイルの諸情報は、iノード6に記録されていることが示されている。第2のエントリには、ファイル"FILE4"が登録されており、そのファイルの諸情報は、iノード7に記録されていることが示されている。

【0042】これを受けて、iノード6には、直接ポインタにより、FILE1のデータが論理ブロック8に記録されていることが示されている。また、iノード7には、直接ポインタにより、FILE2のデータが論理ブロック9に記録されていることが示されている。

【0043】iノードによるファイルシステムにデータを記録するときは、図42と図43のフローチャートに示す手順で行う。

【0044】最初にステップS361で、iノードリストを走査し、参照フラグ(図40)が0の未使用iノードを得る。次にステップS362で、そのiノードに、ファイルのモード、タイムスタンプ、ファイルサイズを記録し、参照フラグを1(使用済)とする。さらにステップS363で、ビットマップを走査し、ビットが1の未使用論理ブロックを得る。

【0045】ステップS364では、その論理ブロックに対応するビットマップのビットを0(使用済)にし、ステップS363で確保した論理ブロックの番号を、ステップS361で得たiノードの直接ポインタに記録する。ステップS365で、さらに論理ブロックが必要と判定されたときは、ステップS366で、ビットマップを走査し、ビットが1の未使用論理ブロックをさらに得る。次にステップS367で、そのビットマップのビットを0(使用済)にし、ステップS366で確保した論理ブロックの番号を、ステップS361で得たiノードの間接ポインタに記録し、間接ブロックとする。

【0046】ステップS368では、ビットマップを走査し、ビットが1の未使用論理ブロックを得る。次にステップS369で、そのビットマップのビットを0(使用済)にし、ステップS368で確保した論理ブロックの番号を、ステップS367で得た間接ブロックに記録する。

【0047】ステップS370で、さらに論理ブロックが必要と判定されたときは、ステップS368に戻り、

必要な数の未使用論理ブロックを得るまで、ステップS368とステップS369を繰り返す。

【0048】ステップS365あるいはステップS370で、必要な数の論理ブロックが得られたら、ステップS371に進み、確保した各論理ブロックに、データを記録する。さらにステップS372に進み、ディレクトリファイルのエントリに、ファイル名とiノード番号を記録する。このディレクトリファイルの記録は、ステップS361乃至S371と同様の処理により行う。

【0049】iノードによるファイルシステムからデータを再生するときは、図44のフローチャートに示す手順で行う。

【0050】ステップS381では、ディレクトリファイル(図41)を走査し、目的のファイル名と一致するディレクトリエントリを見つける。次にステップS382で、見つかったディレクトリエントリからiノード番号を得、iノードリストからそのiノードを読み出す。

【0051】さらにステップS383で、そのiノードの直接ポインタから、ブロック番号を得、その番号の論理ブロックからデータを読み出す。次にステップS384で、間接ポインタから、ブロック番号を得、その番号の論理ブロック(間接ブロック)を読み出す。ステップS385では、その間接ブロックに記述されている直接ポインタから、ブロック番号を得、その番号の論理ブロックからデータを読み出す。

【0052】iノードによるファイルシステムからデータを消去するときは、図45と図46のフローチャートに示す手順で行う。

【0053】ステップS401では、ディレクトリファイルを走査し、目的のファイル名と一致するディレクトリエントリを見つける。ステップS402では、見つかったディレクトリエントリからiノード番号を得、iノードリストからそのiノードを読み出す。

【0054】次にステップS403で、そのiノードの直接ポインタから、ブロック番号を得、ビットマップのそのビットを1(空き)にする。ステップS404では、そのiノードの間接ポインタから、ブロック番号を得、その論理ブロック(間接ブロック)を読み出す。ステップS405では、間接ブロックから、ブロック番号を得、ビットマップのそのビットを1(空き)にする。さらにステップS406で、間接ブロックに対応する、ビットマップのビットを1(空き)にする。

【0055】次にステップS407で、iノードの参照フラグを0(未使用)にする。さらにステップS408で、ディレクトリファイルから、そのディレクトリエントリを削除する。

【0056】実際のUNIXファイルシステムでは、論理ブロックをさらに小さいフラグメントに分割したり、iノードに2重間接、3重間接のポインタを設けるなど、さらに複雑な構成としているが、ここでは省略す

る。

【0057】次に、第3の従来技術の例として、近年オーディオ用として実用化されたMD(ミニディスク)の構成を説明する。

【0058】MDに記録されるデータは、トラックという概念で管理され、これは、前記2例のファイルに相当する。トラックは、必ずしも連続領域に記録する必要はなく、複数の連続領域の集まりでもよい。この各連続領域をパーツと呼ぶ。

【0059】図47は、MDのデータ記録媒体であるMDディスクの構成を表している。MDディスクは、それぞれ74752バイトよりなる、0乃至N-1のN個のクラスタで構成され、さらに各クラスタは、2336バイトよりなる32個のセクタで構成されている。尚、上記クラスタ数におけるNの典型的な値は、約2200である。

【0060】上記クラスタのうち、クラスタ3乃至クラスタ49は、UTOC領域とされ、特にクラスタ3のセクタ0は、ディスク管理情報が記録される領域とされ、UTOCセクタ0と称される。また、クラスタ50以降はユーザー記録領域であり、各ファイルのデータは、この領域に記録される。

【0061】図48は、UTOCセクタ0の構成を表している。このセクタは、FirstTNO、LastTNO、P-Empty、P-FRA、P-TNO1乃至255、パーツディスクリプタ1乃至255により構成される。

【0062】FirstTNOには、P-TNOのうち、使用済の最も小さいP-TNO番号を記録し、LastTNOには、使用済の最も大きいP-TNO番号を記録している。但し、使用済のP-TNOがないときは、FirstTNO、LastTNOともに、0が記録される。

【0063】P-Emptyには、使用されていないパーツディスクリプタ番号のうちの最初の1つが記録され、後述の方法(Link-P)により、使用されていない全てのパーツディスクリプタは、P-Emptyに連結される。

【0064】P-FRAには、記録されていない領域(パーツ)を指すパーツディスクリプタ番号のうちの最初の1つが記録され、後述の方法(Link-P)により、使用されていない領域を指す全てのパーツディスクリプタは、P-FRAに連結される。

【0065】各P-TNOには、トラック(ファイル)のデータが記録された領域を指すパーツディスクリプタのうち、最初のパーツディスクリプタ番号が記録され、後述の方法(Link-P)により、そのトラックのデータが記録された領域を指す全てのパーツディスクリプタは、各P-TNOに連結される。

【0066】パーツディスクリプタは、開始アドレス、

終了アドレス、トラックモード、Link-Pで構成されている。

【0067】開始アドレスには、データが記録されたパーツの最初のセクタのアドレスが記録され、終了アドレスには、そのパーツの最後のセクタのアドレスが記録される。トラックモードには、書込禁止/許可、モノラル/ステレオ等の、データの種別情報が記録される。また、Link-Pには、トラックが複数パーツで構成された場合の、次のパーツディスクリプタ番号が記録され、もしトラックの最終パーツならば、0が記録される。

【0068】上記構成によるMDにデータを記録するときは、図49のフローチャートに示す手順で行う。

【0069】最初にステップS421で、P-FRAにリンクされたパーツから、未記録パーツを得る。即ち、P-FRAに記録されている第1のパーツディスクリプタを得、そのLink-Pの値（次の空きパーツのパーツディスクリプタ番号）をP-FRAに記録することにより、1つの未記録パーツを得る。この処理は、ステップS422で、必要な容量が得られたと判定されるまで繰り返される。

【0070】ステップS423では、確保されたパーツのうちの最後のパーツが、必要サイズより大きいかな否かを判定し、YESのとき、ステップS424に進み、パーツを分割する。即ち、P-Emptyに記録されている第1の（空きの）パーツディスクリプタを得、そのLink-Pの値（次の空きパーツディスクリプタ番号）をP-Emptyに記録することにより、未使用パーツディスクリプタを得、最後のパーツを分割して新たに得られたパーツのアドレス（開始アドレスと終了アドレス）を、このパーツディスクリプタに記録する。そして、最後のパーツ（分割して残されたパーツ）のパーツディスクリプタの終了アドレスを所定値に変更する。

【0071】次にステップS425で、LastTNOを1増加し、得られたトラック番号のP-TNOに、ステップS421で得られた未記録パーツを連結する。そしてステップS426に進み、そのパーツに、データを記録する。

【0072】ステップS423で、パーツが必要サイズより大きくないと判定された場合、ステップS424、S425の処理は、スキップされる。

【0073】上記構成によるMDからデータを再生するときは、図50のフローチャートに示す手順で行う。

【0074】最初にステップS441で、指定されたトラック番号のP-TNOから、第1のパーツディスクリプタを得、そこに記録されている開始アドレスから終了アドレスまでを再生する。次にステップS442で、そのパーツディスクリプタのLink-Pが0かな否かを判定し、0でなければ、ステップS443に進み、次のパーツディスクリプタに記録されている開始アドレスから

終了アドレスまでを再生する。以上の処理を繰り返し、ステップS442で、Link-Pが0と判定されたとき、処理を終了する。

【0075】上記構成によるMDからデータを消去するときは、図51のフローチャートに示す手順で行う。

【0076】最初にステップS461で、指定されたトラック番号のP-TNOから、パーツディスクリプタを得る。次にステップS462に進み、得られたパーツディスクリプタに記録されているパーツを、P-FRAに連結する。即ち、P-FRAに連結されている各パーツディスクリプタを走査し、最後のパーツディスクリプタのLink-Pに、ステップS461で得たパーツディスクリプタ番号を記録し、連結されたパーツディスクリプタのLink-Pは0とする。

【0077】そのトラックに連結されていたパーツを全てP-FRAに連結したとステップS463で判定されるまで、ステップS461、S462の処理を繰り返した後、ステップS464に進み、その（消去対象とされる）P-TNOより後の全てのP-TNOを1つ前に移動して、そのP-TNOを消去し、LastTNOを1減じる。

【0078】

【発明が解決しようとする課題】ところで、上述した従来技術には、以下のような問題点があった。

【0079】前記第1の従来例のMS-DOS及び第2の従来例のUNIXでは、次のような問題点がある。

【0080】（1）各データの記録に際し、ファイル名が定義されることを前提にしているため、デジタルスチルカメラなど、キーボードを持たない機器への適用が困難である。

【0081】（2）ファイルの階層的管理のためにディレクトリファイルを必要とするなど、管理情報が大きく、メモリ容量に余裕のない機器への適用が困難である。

【0082】（3）ディレクトリファイルがデータ記録領域に記録され、またUNIXにおいては、iノードの間接ブロックもデータ記録領域に記録されるなど、ファイル管理情報がデータ記録領域に散在することになり、シークの遅いディスク等では、速度低下の問題が生じる。

【0083】（4）ファイルの生成や消去に伴って、ディレクトリファイルの更新や拡張が必要となり、速度低下の問題が生じる。

【0084】一方、第3の従来例のMDにおいては、ファイル名が不要なため、上記（1）の問題はなく、また管理情報が小さいため、上記（2）乃至（4）の問題もない。しかしながら、以下のような問題点があった。

【0085】（5）ファイルを階層的に管理する機構がない。

【0086】（6）パーツ数が、255個に限られてお

り、多数のファイルを管理できない。

【0087】(7) 本来は各トラックごとの情報であるトラックモードが、全てのパーティションディスクに記録されるなど、管理情報の効率が悪い。

【0088】本発明はこのような状況に鑑みてなされたものであり、これらの問題点を解消することができるようにするものである。

【0089】

【課題を解決するための手段】本発明のファイル管理方法は、多数のブロックに区分した領域にファイルを記録し、再生するようにした記録媒体のファイルを管理するファイル管理方法において、記録媒体に、ブロックの使用状態を表す使用情報を記録するブロック管理領域（例えば図2のビットマップセクタ0乃至3）を形成するとともに、ファイルの記録に使用されたブロックに関するインデックス情報を記録するインデックス領域（例えば図2のインデックスセクタ0乃至250）とを形成し、インデックス領域に、ファイルに関連づける関連情報を記録することを特徴とする。

【0090】この関連情報は、ファイルをグループに分類して管理するグループタグ、親のファイルを特定する番号を記録するインデックスタグ、あるいは、連続するインデックス領域を特定する情報を記録するリンクタグを含むようにすることができる。

【0091】リンクタグに連続するインデックス領域を特定する情報が記録されているか否かを表すリンクフラグをさらに関連情報に含ませることもできる。さらに、1つのファイルを管理する複数のインデックス領域のうち、先頭のインデックス領域であるか否かを表す拡張フラグを含ませることもできる。

【0092】また、複数のインデックス領域に記録された情報によりインデックスページを形成し、インデックスページの1ページ分の情報を記憶することができるページバッファメモリを複数ページ分設け、ページバッファメモリを管理する情報を記憶するページタグを形成し、ページタグに、インデックスページを特定する番号、ページバッファメモリを指定するポインタ、ページバッファメモリの内容が更新されたか否かを表す更新情報の少なくとも1つを記録するようにすることができる。

【0093】また、記録媒体に、インデックスページにおける未使用インデックス領域の数を表すページアカウンタを記録するページアカウンタ領域を形成することができる。この場合、記録媒体のページアカウンタ領域より読み出したページアカウンタを記憶するページアカウンタテーブルをメモリに形成するようにすることができる。

【0094】さらに、インデックスブロックには、属性コードとして、ディレクトリファイルであるか否かを表すコードを記録し、ディレクトリファイルのデータとし

て子ファイルのインデックス番号を記録し、子ファイルのインデックスブロックには、インデックスタグとして、ディレクトリファイルのインデックス番号を記録することができる。

【0095】また、ディレクトリファイルのデータとして、子ファイルのインデックス番号とともに、子ファイルの名称を記録したり、属性コードを、複数の種類のディレクトリファイルのいずれかを表す複数ビットのコードで構成することができる。

【0096】

【作用】上記構成のファイル管理方法においては、インデックス領域にファイルに関連付ける関連情報が記録される。従って、ファイル名がなくとも、多数のファイルを階層的に管理することが可能になる。また、管理情報の効率化を図ることができる。

【0097】

【実施例】

〈実施例1〉以下、図面を参照し、本発明の実施例について説明する。

【0098】尚、本発明のファイル管理方法が適用されるデータ記録再生装置は、図32に示した従来の場合と基本的に同様の構成を有しているため、その説明は省略するが、ファイル管理の仕方が従来の場合と異なっている。

【0099】図1は、本実施例で使用するデータ記録媒体（MDディスク（データ））1（以下、必要に応じて、単にMDデータと記す）の物理的な構成図（フォーマット）を表している。MDデータは、65536バイトよりなる0乃至N-1のN個のクラスタで構成されている。現時点における、Nの典型的な値は約2200であり、また、将来にわたってさらなる大容量化が期待できる。上記各クラスタは、それぞれ2048バイトよりなる0乃至31の32個のセクタで構成されている。

【0100】このMDデータに対する書き込みは、クラスタを最小単位として行う必要があり、読み出しは、セクタを最小単位として行う必要がある。

【0101】本実施例においては、ファイルへの領域割当の最小単位をセクタ4個（勿論、それ以上、またはそれ以下とすることも可能である）により構成し、割当ブロックと名付ける。また、全クラスタ0乃至N-1における割当ブロックに通し番号を付け、割当ブロック番号とする。

【0102】図2は、MDデータの論理的な構成図である。クラスタ0, 2, 4, 6, 8, 10, 12, 14の計8クラスタは、管理クラスタとされ、ファイルの管理情報を記録する。また、クラスタ1, 3, 5, 7, 9, 11, 13, 15は予備管理クラスタとされ、管理クラスタと同一の情報を記録し、万一のデータ喪失に備える。残りのクラスタ16乃至N-1は、全てデータの記録領域として使用する。

【0103】管理クラスタ0乃至7を構成する各セクタを管理セクタと称する。管理セクタ0は、ボリューム情報セクタとされ、割当ブロックサイズ、ディスク容量等の、各種パラメータが記録される。

【0104】管理セクタ1, 2, 3, 4は、ビットマップセクタ（ビットマップセクタ0乃至3）とされ、アロケーションマップが記録される。アロケーションマップは、各割当ブロックにつき各1ビットよりなる、全65536ビット（=2048バイト×4）で構成され、ビットが0のとき、その割当ブロックが使用可能であり、ビットが1のとき、使用済であることを表す。従って、65536個の割当ブロック、即ち、8192クラスタ（=512M（メガ）バイト）まで管理可能である。但し、数値\$FFFFは、空き判定等に使用するため、ブロック番号としての使用が禁止される。その結果、管理可能な最大割当ブロック数は、65535個となる。

【0105】管理セクタ5乃至31, 64乃至95, 128乃至159, ...は、インデックスセクタ0乃至250とされ（管理セクタ37乃至63, 96乃至127, ...は、予備のインデックスセクタ0乃至250とされ）、各ファイルの管理情報を記録する。

【0106】各インデックスセクタは、各16バイトよりなる128個のインデックスブロック（以下、IBと呼ぶ）により構成される。インデックスセクタ0乃至250における計32128個のIBには、0乃至32127のIB番号が付けられる。

【0107】図3は、IBの構成例を示している。その第1の要素は、グループタグであり、2バイトで構成される。このグループタグの値が\$FFFFのとき、IBが空きであることを示す。第2の要素は属性コードであり、2バイトで構成される。第3の要素は更新日時等のタイムスタンプであり、4バイトで構成される。第4の要素はファイルサイズであり、4バイトで構成される。第5の要素はデータの位置を示すエクステント記述子であり、4バイトで構成される。

【0108】エクステント記述子は、割当ブロック数と開始割当ブロック番号により構成される。2バイトで示される割当ブロック数は、ファイルデータが記録された領域の大きさを割当ブロックの数で示し、2バイトの開始割当ブロック番号は、ファイルデータの記録開始位置（ファイルの先頭の割当ブロックの番号）を示す。

【0109】上述したMS-DOSにおけるディレクトリエントリのうち、ファイル名を除く全ての要素を、16バイトのIBに保持することができる。

【0110】このようなMDデータに、データを記録するときは、図4と図5のフローチャートに示す手順で行なう。

【0111】最初にステップS1で、記録媒体1上の管理クラスタ0乃至7（図2）の内容を、メモリ3上に読み込む。次にステップS2で、メモリ3上のインデック

スセクタ0乃至250を走査し、ステップS3で、空きIB（グループタグ=\$FFFFのIB）があるか否かを判定し、あると判定されたら、ステップS5に進み、グループタグ（例えば、その属するグループの番号）と属性コード（例えば、ディレクトリ属性なら1、データ属性なら0）を記録して、IBを確保する。空きIBがなければ、ステップS4に進み、エラーとして終了する。

【0112】次にステップS6で、メモリ3上のアロケーションマップ（ビットマップセクタ0乃至3）（図2）を走査し、ステップS7で、必要な個数の連続する空き（グループタグ=\$FFFFの）割当ブロックを見つけたか否かを判定し、見つかったら、ステップS8に進み、そのビットを1（使用済）にして、確保する。見つからなければ、ステップS4に進み、エラーとして終了する。

【0113】次にステップS9に進み、ステップS8で確保したIBに、ステップS7で確保した割当ブロックの個数と、そのうちの開始割当ブロックの番号を記録する。さらにステップS10に進み、ステップS8で確保した記録媒体1の領域（割当ブロック）に、データを記録する。次にステップS11に進み、ステップS5で確保したメモリ3上のIBに、タイムスタンプとファイルサイズを記録する。また、ステップS12において、メモリ3上の管理クラスタ0乃至7の内容を、記録媒体1上の管理クラスタ0乃至7と、予備管理クラスタ0乃至7に書き込む。

【0114】このようにしてファイルが記録されると、以後、IB番号が、記録したデータのファイル識別子とされる。

【0115】次に、データを再生するとき、図6のフローチャートに示す手順で行なう。

【0116】最初にステップS31で、再生するファイルのIB番号を含むインデックスセクタを、記録媒体1からメモリ3上に読み込む。次にステップS32で、そのIBより、開始割当ブロック番号と割当ブロックの個数を得る。そして、ステップS33で、記録媒体1上のその開始割当ブロックから所定の個数の割当ブロックのデータを再生する。

【0117】また、データを消去するとき、図7のフローチャートに示す手順で行なう。

【0118】最初にステップS51で、記録媒体1上の管理クラスタ0乃至7を、メモリ3上に読み込む。次にステップS52で、消去するファイルのIBを得、開始割当ブロック番号と割当ブロックの個数を得る。そして、ステップS53で、メモリ3上のアロケーションマップにおいて、その割当ブロックに相当するビット列を0（使用可能）にし、領域を解放する。

【0119】さらに、ステップS54で、IBのグループタグに\$FFFFを書き、空きIBとし、ステップS

55で、メモリ3上の管理クラスタ0乃至7の内容を、記録媒体1上の管理クラスタ0乃至7と、予備管理クラスタ0乃至7に書き込む。

【0120】以上のようにして、インデックスセクタ0乃至250により作成されたインデックステーブルの例を、図8に示す。図8では、以下の説明に関係しない要素について、一で表し、記述を省略している。

【0121】図8に示すように、IB番号0のファイルは、グループ0（グループタグ=0）に属し、65536（64k）バイトのデータよりなる。このデータは、割当ブロック0乃至7の8個の割当ブロックに記録されている。

【0122】IB番号1のファイルは、グループ0に属し、131072（128k）バイトのデータよりなる。このデータは、割当ブロック8乃至23の16個の割当ブロックに記録されている。

【0123】IB番号2のファイルは、グループ1に属し、8192（8k）バイトのデータよりなる。このデータは、割当ブロック24の1個の割当ブロックに記録されている。

【0124】IB番号3のファイルは、グループ2に属し、131072（128k）バイトのデータよりなる。このデータは、割当ブロック25乃至40の16個の割当ブロックに記録されている。

【0125】IB番号4のファイルは、グループ1に属し、1048576（1024k）バイトのデータよりなる。このデータは、割当ブロック41乃至168の128個の割当ブロックに記録されている。

【0126】IB番号5のファイルは、グループ1に属し、8192（8k）バイトのデータよりなる。このデータは、割当ブロック169の1個の割当ブロックに記録されている。

【0127】IB番号6以降のIBは、すべて空きIBであり、グループタグには、\$FFFFが記録されている。

【0128】上記グループタグにより、
グループ0=ファイル0（IB番号0）、ファイル1（IB番号1）
グループ1=ファイル2（IB番号2）、ファイル4（IB番号4）、ファイル5（IB番号5）
グループ2=ファイル3（IB番号3）
のように、各ファイルをグループに分類して記録することが実現できる。

【0129】次に、図9に、上記グループタグをインデックスタグに代え、インデックスタグに他のIB番号を記録して、階層的管理を行うようにしたインデックステーブルの例を示す。

【0130】図9に示すように、IB0には、インデックスタグに0（自分自身）が記録されており、ルートディレクトリであることが表されている。また、ファイル

属性には1が記録されており、ディレクトリ属性であることが表されている。また、ファイルサイズには0が記録されており、ファイルの実体データがないことが表されている。

【0131】IB1には、インデックスタグに0が記録されており、IB0を親ディレクトリとするファイルであることが表されている。また、ファイル属性には0が記録されており、データ属性のファイルであることが表されている。また、ファイルサイズは131072バイトとされ、また、そのデータが割当ブロック0から、16個の割当ブロックにわたって記録されていることがわかる。

【0132】IB3には、インデックスタグに0、ファイル属性に1が記録されており、IB0を親ディレクトリとするサブディレクトリファイルであることが表されている。また、ファイルサイズには0が記録されており、ファイルの実体データがないことが表されている。

【0133】IB4には、インデックスタグに3、ファイル属性に0が記録されており、IB3を親ディレクトリとするデータファイルであることが表されている。また、ファイルサイズは8192バイトであり、そのデータが割当ブロック24に記録されていることがわかる。

【0134】IB番号6以降のIBは、すべて空きIBであり、インデックスタグには\$FFFFが記録されている。

【0135】このように、インデックスタグに親ディレクトリを記述することにより、図10に示すように、ファイル（IB）を階層構造で管理することができる。このように、親ディレクトリを記述しておくこと、そのファイルを作成したとき、親ディレクトリの更新が不要となる。

【0136】尚、図9の例では、ディレクトリファイルのサイズを0としたが、ディレクトリファイルはデータを持ってもよく、これを利用して、複数キーワードを許すデータベースを記録するなど、従来のファイルシステムにはない高度な管理構造を実現することもできる。

【0137】〈実施例2〉上記実施例1におけるIBの構成は、各IBについて、1つのエクステンツ記述子しか記録できない。然るに、一般のデータ記録再生装置において、可変長データの記録、消去、追記などを繰り返すと、1つのファイルが複数の記録領域（割当ブロック）により構成される可能性がある。

【0138】そこで、実施例1のアロケーションマップによる領域管理に代えて、上述したMS-DOSの場合と同様に、FATによる領域管理を採用し、エクステンツ記述子として、開始割当ブロック番号を記録することが考えられる。

【0139】しかしながら、そのようにすると、実施例1におけるアロケーションマップのサイズが、4セクタ=8192バイトであったのに対し、同等のFATを構

成するには、131070バイト必要であり、必要メモリ量が大きくなってしまふ。

【0140】そこで、さらに別の解決法として、上述したUNIXの例にならい、間接ブロックを用いて複数のエクステント記述子を記録することが考えられる。

【0141】しかしながら、そのようにすると、例えば、新たに得たエクステント記述子が1個（4バイト）であったとしても、1つの割当ブロック＝8192バイトを消費するため、容量効率が悪くなり、またエクステント記述子がデータ記録領域に散在する可能性が高いため、シーク動作の遅いディスク装置等では速度低下の問題がある。

【0142】このようなことから、実施例2においては、実施例1におけるIBに、以下の拡張を施す。

【0143】図11は、IBの基本型を表す。第1の要素は拡張フラグであり、1ビットで構成される。拡張フラグが1であることにより、基本型のIBであることを示し、0であることにより、拡張型のIB（図12）であることを示す。

【0144】第2の要素はインデックスタグであり、15ビットで構成される。第3の要素は属性コードであり、16ビットで構成される。第4の要素は更新日時のタイムスタンプであり、4バイトで構成される。第5の要素はファイルサイズであり、4バイトで構成される。

【0145】第6の要素はリンクフラグであり、1ビットで構成される。リンクフラグが0のとき、第7の要素はエクステント記述子であり、31ビットで構成される。リンクフラグが1のとき、第7の要素はリンクタグであり、15ビットで構成され、余白の16ビットは保留として0を記録する。

【0146】図12は、IBの拡張型を表す。第1の要素は拡張フラグであり、1ビットで構成される。拡張フラグが0であることにより、拡張型のIBであることを示す。第2の要素はエクステント記述子であり、31ビットで構成される。

【0147】第3の要素は保留ビットであり、0が記録される。第4の要素はエクステント記述子であり、31ビットで構成される。

【0148】第5の要素は保留ビットであり、0が記録される。第6の要素はエクステント記述子であり、31ビットで構成される。

【0149】第7の要素はリンクフラグであり、1ビットで構成される。リンクフラグが0のとき、第8の要素はエクステント記述子であり、31ビットで構成される。リンクフラグが1のとき、第8の要素はリンクタグであり、15ビットで構成され、余白の16ビットは保留として0を記録する。

【0150】図13は、エクステント記述子の構成例を示す。15ビットで示される割当ブロック数により、ファイルデータが記録された領域の大きさを示し、16ビ

ットの開始割当ブロック番号により、ファイルデータの記録開始位置を示す。

【0151】この図13に示したIBの構成により、IBを拡張した例を、図14に示す。この図は、ファイル識別子（IB）＝4として記録されたファイルの例を表している。

【0152】IB4の拡張フラグは1であり、IB4が基本型IBであることを表している。インデックスタグは3であり、IB＝3のディレクトリを親とするファイルであることを表す。ファイル属性は0であり、データファイルであることを表す。

【0153】ファイルサイズは581632であり、568kバイト＝71割当ブロックのサイズであることを表す。リンクフラグは1であり、他のIBに拡張されたことを表す。リンクタグは6であり、拡張された次のIBは、IB6であることを表す。

【0154】拡張された、IB6の拡張フラグは0であり、IB6が拡張型IBであることを表している。第1のエクステント記述子は、割当ブロック80から、8ブロックにわたってファイルデータが記録されていることを表す。第2のエクステント記述子は、割当ブロック104から、16ブロックにわたってファイルデータが記録されていることを表す。第3のエクステント記述子は、割当ブロック176から、8ブロックにわたってファイルデータが記録されていることを表す。リンクフラグは1であり、IB4が、IB6からさらに他のIBに拡張されたことを表す。リンクタグは13であり、拡張された次のIBは、IB13であることを表す。

【0155】拡張された、IB13の拡張フラグは0であり、IB13が拡張型IBであることを表している。第1のエクステント記述子は、割当ブロック216から、24ブロックにわたってファイルデータが記録されていることを表す。第2のエクステント記述子は、割当ブロック248から、15ブロックにわたってファイルデータが記録されていることを表す。リンクフラグは0であり、ファイル4は、このIBで終結することを表す。

【0156】以上説明したように、拡張フラグ、リンクフラグおよびリンクタグを用いれば、空きIBがあるかぎり、ファイルにおけるエクステント記述子の個数を自由に拡張することができる。

【0157】〈実施例3〉ところで、実施例1において、データの記録、再生、消去の動作を行うには、上述したように、クラスタ0、2、4、6、8、10、12、14の内容（管理クラスタ0乃至7の内容）をメモリ3上に持つ必要があり、512kバイトもの大量のメモリを消費することになる。

【0158】この問題を解決するため、実施例3においては、以下のようにして、図15に示すようなIBキャッシュシステムを設ける。

【0159】(1) 記録媒体1上のインデックステーブルを、128個のIBよりなるインデックスページにより構成する。即ち、1管理セクタをもって1インデックスページとし、251インデックスページによりインデックステーブルを構成する(図2)。

【0160】(2) メモリ3上に、各インデックスページにつき1バイトのページアカウントよりなる、251バイトで構成されるページアカウントテーブルを設ける。各ページアカウントには、そのインデックスページ中の、空きIBの個数が記録される。ページアカウントテーブルの内容は、記録媒体1上において、ボリューム情報の1つとして、管理セクタ0に記録される(図2)。

【0161】(3) メモリ3上に、それぞれインデックスページ1個分を保持できる、4個のページバッファメモリを設ける。

【0162】(4) メモリ3上に、インデックスページ番号、ページバッファメモリを指すポインタ、およびページバッファメモリの内容が更新されたことを示す更新フラグにより、ページタグを構成し、ページタグ4個により、ページタグテーブルを構成する。

【0163】(5) メモリ3上に、65536バイト(1クラスサイズ)よりなる書き込みバッファを設ける。

【0164】図16は、ページタグテーブルの構成例を示す。ページタグテーブルは、0乃至3の4つのページタグで構成され、各ページタグは、1バイトよりなる更新フラグ(この更新フラグは、記録媒体1からメモリ3(ページバッファメモリ)に読み出したデータが、メモリ3上において更新されたとき、1とされ、更新されないとき(記録媒体1のデータと同一のデータであるとき)、0とされる)と、1バイトよりなるインデックスページ番号と、4バイトよりなるページバッファポインタで構成される。

【0165】上記構成において、記録媒体1上のインデックスページNの内容を、メモリ3上のページバッファメモリに読み出すには、図17のフローチャートに示す手順で行なう。

【0166】最初にステップS91で、変数iに0を初期設定した後、ステップS92で、ページタグ0のインデックスページ番号を読み出し、ステップS93で、その値を判定する。その値がNと判定されたとき、インデックスページNがページバッファメモリ0に既に読み込まれているので、処理を終了する。インデックスページ番号がNでないとき、ステップS94で、\$FF(未使用)か否かを判定し、YESならば、ステップS95に進み、ページタグ0のインデックスページ番号にNを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体1上のインデックスページNを読み出し、処理を終了する。

【0167】ステップS94で、インデックスページ番号が\$FFではない(使用済)と判定されたとき、ステップS96で、変数iを判定し、変数iが3でないと判定されれば、ステップS97で、変数iを1だけインクリメントし($i=1$ とし)、ステップS98で、ページタグ0とページタグi(いまの場合、 $i=1$)を交換する。

【0168】そして、ステップS92に戻り、交換したページタグ0のインデックスページ番号を読み出し、ステップS93でNと判定されたら終了する。番号がNでなければ、ステップS94で\$FF(未使用)か否かを判定し、ページタグ0のインデックスページ番号が\$FF(未使用)ならば、ステップS95に進み、そのインデックスページ番号にNを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体1上のインデックスページNを読み出し、処理を終了する。

【0169】インデックスページ番号が\$FFでないとき(使用済のとき)、ステップS96で、 $i=3$ でなければ(いまの場合、 $i=1$)、ステップS97で、iを1だけインクリメントして、 $i=2$ とした後、ステップS98で、ページタグ0とページタグi(いまの場合、 $i=2$)を交換する。

【0170】再びステップS92に戻り、交換したページタグ0のインデックスページ番号を読み出し、Nならば、処理を終了する。また、ステップS94で、ページタグ0のインデックスページ番号が\$FF(未使用)と判定されたならば、ステップS95で、そのインデックスページ番号にNを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体1上のインデックスページNを読み出し、処理を終了する。

【0171】さらに、インデックスページ番号が\$FFではなく(使用済であり)、iが3でなければ(いまの場合、 $i=2$)、iを1だけインクリメントして、 $i=3$ とし、ページタグ0とページタグi(いまの場合、 $i=3$)を交換する。

【0172】ステップS92で、再び読み取ったページタグ0のインデックスページ番号がNならば、処理を終了する。そのページタグ0のインデックスページ番号が\$FF(未使用)ならば、ステップS95で、そのインデックスページ番号にNを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体1上のインデックスページNを読み出し、処理を終了する。

【0173】ステップS94とステップS96で、インデックスページ番号が\$FFではなく(使用済であり)、かつ、 $i=3$ と判定されたとき、結局、ページバッファ0乃至3の4個が全て使用済であることになる。この場合、ステップS99に進み、ページタグ0の更新フラグが1(対応するページバッファメモリのデータを更新した)か否かを判定し、1ならば、ステップS100

に進み、ページタグ0のページバッファポインタの指すページバッファメモリのデータ（更新データ）を、そのインデックスページ番号の示すインデックスページ（記録媒体1）に書き込み、その更新フラグを0にする。即ち、更新データを保存する。

【0174】次にステップS101に進み、ページタグ0のインデックスページ番号にNを書き込み、そのページバッファポインタの指すページバッファメモリに、記録媒体1上のインデックスページNの内容を読み出し、処理を終了する。

【0175】ステップS99で、更新フラグが0（データを更新していない）と判定されたとき、保存処理は不要となるので、ステップS100の処理はスキップされ、ステップS101の処理が直ちに実行される。

【0176】この手順による具体例として、図18および図19を参照し、ページタグテーブルの動作について、さらに説明する。

【0177】図18（a）は、ページタグテーブルの初期状態を示す。メモリ3上には、4つのページバッファメモリが、それぞれアドレス\$10000、\$10800、\$11000、\$11800に置かれており、各ページタグのページバッファポインタは、この各アドレスを指している。また、各インデックスページ番号は、\$FFであり、未使用ページタグであることを示す。

【0178】図18（b）は、インデックスページ0を読み出した状態を表している。ページタグ0のインデックスページ番号には0が書かれ、メモリ3上のアドレス\$10000（ページバッファメモリ0）に、記録媒体1からインデックスページ0の内容が読み込まれる。

【0179】図18（c）は、さらに、インデックスページ1を読み出した状態を表している。ページタグ0とページタグ1が交換され、ページタグ0のインデックスページ番号には1が書かれ、メモリ3上のアドレス\$10800（ページバッファメモリ1）に、インデックスページ1の内容が読み込まれる。

【0180】図18（d）は、同様にして、インデックスページ2および3を読み出した状態を表している。同図に示すように、若いページタグには、より新しいインデックスページが順次記録される。

【0181】図19（a）は、ページタグ3と1の更新フラグが1であり、インデックスページ0と2の内容が更新された状態を表している。

【0182】図19（b）は、インデックスページ4を読み込む途中（図18（c）に至るまでの途中）経過を表している。最も古いページタグ3（図18（a））をページタグ0に移し、そのインデックスページ0の内容（更新されている）を記録媒体1に書き込んで、更新フラグをクリアする。このとき、後の書き込み動作を節約するため、同一クラスタに属するインデックスページ2も、同時に書き込み、更新フラグをクリアする。

【0183】図19（c）は、インデックスページ4を読み込んだ状態を表している。ページタグ0のインデックスページ番号には4が書かれ、メモリ3のアドレス\$10000に、インデックスページ4の内容が読み込まれる。

【0184】図19（d）は、再びインデックスページ2を読み出した状態を表している。既にページバッファメモリに読み出されているため、実際の読み出し動作は行なわれず、最新にアクセスされたページとして、ページタグ2がページタグ0に移されるだけであり、高速に動作する。

【0185】図17のステップS100において、ページタグ0のページバッファポインタが指すページバッファメモリのデータを、そのインデックスページ番号の示すインデックスページ（記録媒体1）に書き込むとき、図20のフローチャートに示す手順で行なう。

【0186】最初にステップS121で、ページタグ0のインデックスページ番号（0乃至250）から、クラスタ番号C0とセクタ番号S0を求める。即ち、この実施例の場合、1インデックスページを1セクタとしているため、インデックスページ番号は、インデックスセクタ番号（0乃至250）と等しい。このインデックスページ番号をNとすると、ボリューム情報セクタとビットマップセクタ0乃至3の合計5セクタを加味して、クラスタ番号C0は、 $(N+5) \div 32$ の商を、2倍して（予備管理クラスタのため）得られる。また、 $(N+5) \div 32$ の余りが、セクタ番号S0となる。

【0187】例えば、インデックスページ27は、 $(27+5) \div 32 = 1$ （余り0）であり、クラスタ番号=2、セクタ番号=0となる。

【0188】次にステップS122で、記録媒体1上のクラスタC0のデータを、メモリ3上の書き込みバッファメモリ（1クラスタサイズ）に読み込む。そしてステップS123で、書き込みバッファ中のS0×2048バイトの位置へ、ページタグ0のページバッファポインタが指すページバッファメモリの内容（1セクタ分）をコピーする（図17における場合と同様の処理が行われる）。

【0189】次にステップS124で、変数iを1に初期設定し、ステップS125で、ページタグi（いまの場合、i=1）のインデックスページ番号から、上述した場合と同様にして、クラスタ番号C1とセクタ番号S1を求める。さらにステップS126で、クラスタ番号Ci（いまの場合、Ci=C1）がクラスタ番号C0と等しいか否か、即ち、同一クラスタC0に含まれるか否かを判定し、Ci=C0（同一クラスタ）ならば、ステップS127に進み、書き込みバッファメモリ中のSi×2048（いまの場合、Si=S1）バイトの位置へ、ページタグi（いまの場合、i=1）のページバッファポインタが指すページバッファメモリの内容をコピ

一する。

【0190】 $C_i = C_0$ ではない場合（異なるクラスタの場合）、またはステップS127のコピーが完了したとき、ステップS128に進み、変数 i が3でなければ、さらにステップS129に進み、変数 i を1だけインクリメントして、 $i = 2$ とする。

【0191】そして、再びステップS125に戻り、ページタグ i ($i = 2$) のインデックスページ番号から、クラスタ番号 C_i ($= C_2$) と、セクタ番号 S_i ($= S_2$) を求める。

【0192】そして、 $C_2 = C_0$ ならば、書き込みバッファ中の $S_2 \times 2048$ バイトの位置へ、ページタグ2のページバッファポインタが指すページバッファメモリの内容をコピーする。

【0193】同様に、 i をインクリメントして、 $i = 3$ とし、ページタグ3のインデックスページ番号から、クラスタ番号 C_3 とセクタ番号 S_3 を求める。そして、 $C_3 = C_0$ ならば、書き込みバッファ中の $S_3 \times 2048$ バイトの位置へ、ページタグ3のページバッファポインタが指すページバッファメモリの内容をコピーする。

【0194】ステップS128において、 $i = 3$ と判定されたとき（ページバッファ0乃至3の全ての内容が書き込みバッファメモリに書き込まれたとき）、ステップS130に進み、書き込みバッファメモリのデータを、記録媒体1上のクラスタ C_0 に記録する。

【0195】このようにして、図15に示すように、記録媒体1から書き込みバッファメモリに1クラスタ分のデータが読み込まれ、ページタグの指すページバッファメモリのデータが書き込みバッファメモリの所定のセクタ位置に移され、書き込みバッファメモリの内容が記録媒体1の元のクラスタに書き込まれる。

【0196】ところで、上記構成のIBキャッシュシステムにおいて、空きIBを見つけるために、インデックスページを0から順にメモリ3上に読み出して調べるようにすると、速度低下の原因となる。これをさけるため、この実施例においては、各インデックスページにおける空きIBの個数を記録するページアカウントテーブルが、記録媒体1のボリューム情報セクタ内に設けられている。

【0197】図21は、ページアカウントテーブルの例である。同図に示すように、インデックスページ0のページアカウントは0であり、空きIBがないことを表している。また、インデックスページ1のページアカウントは1であり、空きIBが1個あることを表している。また、インデックスページ2のページアカウントは71であり、空きIBが71個あることを表している。さらに、インデックスページ250のページアカウントは128であり、全てのIBが空きであることを表している。

【0198】基本型IB（図11）を記録するために空

きIBを求める場合は、ページアカウントが2以上のインデックスページを読み込む。なぜなら、基本型IBはエクステント記述子が1個しか記録できないため、すぐに次の空きIBが必要になる可能性が高いが、この場合、そのIBが同一ページ（同一のインデックスセクタ）にあれば、高速化が期待できるからである。

【0199】また、拡張型IB（図12）を記録するために空きIBを求める場合は、1つ前の拡張型IBまたは基本型IBが属するインデックスページのページアカウントから昇順に走査する。なぜなら、それより前に空きIBがあったとしても、それは他のファイルが消去された結果であるため、ランダムな位置にあり、距離が遠くなって、IBキャッシュの効果が減少するからである。

【0200】以上のIBキャッシュシステムにおいて、データを記録するには、図22と図23のフローチャートに示す手順で行う。

【0201】最初にステップS151で、記録媒体1上のページアカウントテーブル（図21）を走査し、ステップS152で、ページアカウントが2以上のインデックスページNが見つかったか否かを判定し、見つからなければ、エラーとして処理を終了する。見つかったときは、ステップS153に進み、記録媒体1上のインデックスページNのデータ（1セクタ分のデータ）を、ページタグ0のポインタが指定するページバッファメモリに読み込む。

【0202】次にステップS154で、そのページバッファメモリを走査して空きIB（インデックスタグ= \$FFFFのIB）を得、そのインデックスタグに、例えば、その上位のIB番号を記録するとともに、属性コードに、ディレクトリ属性なら1、データ属性なら0を記録して、IBを確保し、ページタグ0の更新フラグを1（更新）にする。また、そのページアカウントを1減らす。さらにステップS155で、記録媒体1上のアロケーションマップを走査し、必要な個数の連続する空き（ビットが0の）割当ブロックを見つけ、対応するアロケーションマップのビットを1（使用済）にして、その割当ブロックを確保する。

【0203】ステップS156では、ステップS154で確保したIBに、エクステント記述子として、ステップS155で確保した開始割当ブロック番号と割当ブロックの個数を記録する。ステップS157では、ステップS155で確保した領域（割当ブロック）に、データを記録する（図20に示した場合と同様の処理が行われる）。また、ステップS158で、メモリ3のIBに、タイムスタンプとファイルサイズを記録する。このメモリ3上のIBは、メモリ3上のその記憶領域を他の目的のために使用するとき、装置の電源をオフするとき、記録媒体1をイジェクトするときなど、所定のタイミングで、さらに記録媒体1に記録される。

【0204】尚、以後、このIB番号を、記録したデータのファイル識別子とする。

【0205】IBキャッシュシステムにおいて、データを再生するには、図24と図25のフローチャートに示す手順で行う。

【0206】最初にステップS171で、再生するファイルのIB番号を128（ページバッファメモリに記憶される1ページ分のインデックスページのIBの個数）で割り、その商の整数部から、インデックスページ番号を得る。ステップS172では、そのインデックスページのデータを、ページバッファに読み込む（図17に示した場合と同様の処理が行われる）。そしてステップS173では、ページバッファメモリ中のそのIBから、ファイルサイズを得る。

【0207】次にステップS174において、ファイルサイズを判定し、0ならば、終了する。

【0208】次にステップS175において、リンクフラグを判定し、リンクフラグが0なら、ステップS176に進み、そのエクステント記述子から開始割当ブロックを得、ステップS177において、その開始割当ブロックから上記したファイルサイズのデータを再生し、終了する。ステップS175において、リンクフラグが1なら、ステップS178に進む。

【0209】ステップS178において、リンクタグより次の拡張IB番号を得る。

【0210】次にステップS179で、拡張IB番号を128で割り、その商の整数部から、インデックスページ番号を得る。そしてステップS180で、そのインデックスページのデータを、ページバッファに読み込む（図17に示した場合と同様の処理が行われる）。

【0211】次にステップS181で、変数*i*に0を初期設定する。

【0212】次にステップS182において、第*i*（いまの場合、第0）のエクステント記述子から、割当ブロック数と開始割当ブロック番号を得る。

【0213】次にステップS183において、ファイルサイズとエクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを比較し、エクステントサイズの方が大きい等しければ、ステップS185に進み、開始割当ブロックから上記ファイルサイズのデータを再生し、終了する。

【0214】ステップS183において、ファイルサイズの方が大きければ、ステップS184に進み、開始割当ブロックから上記エクステントサイズのデータを再生する。

【0215】次にステップS186において、ファイルサイズから上記エクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを減じ、また変数*i*をインクリメントして、*i*=1とする。そしてステップS187で、*i*=4でないことを確認し、次にステップS188

で第*i*（いまの場合、第1）のエクステント記述子の最上位ビット（リンクフラグ）を判定し、リンクフラグが0ならば、ステップS182に戻って、次のエクステントを再生し、リンクフラグが1ならば、ステップS178に戻って、次の拡張IBを得る。

【0216】ステップS187において、*i*=4のとき、ファイルサイズが0でないのに次の拡張IBがないことを意味するので、エラーであり、エラー処理を行う。

【0217】さらに、IBキャッシュシステムにおいて、データを消去するには、図26乃至図28のフローチャートに示す手順で行う。

【0218】最初にステップS201で、消去するファイルのIB番号を128で割り、その商の整数部から、インデックスページ番号を得る。そしてステップS202において、そのインデックスページのデータを、ページバッファに読み込む（図17に示した場合と同様の処理を行う）。次にステップS203で、ページバッファメモリ中のそのIBから、ファイルサイズを得、ステップS204で、そのファイルサイズを判定し、0ならば、ステップS205以降のデータ領域の解放処理をスキップし、ステップS208に進む。

【0219】ファイルサイズが0でなければ、ステップS205において、リンクフラグを判定し、リンクフラグが0なら、ステップS206で、そのエクステント記述子から割当ブロック数と開始割当ブロック番号を得、次にステップS207で、アロケーションマップにおいて、上記開始割当ブロック番号のビット位置から、割当ブロック数分のビットを0にして、データ領域を解放する。

【0220】次にステップS208において、そのIBのインデックスタグに\$FFFFを記録して未使用IBとし、ステップS209において、そのIBが属するページバッファを指すページタグの更新フラグをセットし、ステップS210において、そのインデックスページのページアカウンをインクリメントし、終了する。

【0221】ステップS205において、リンクフラグが1（次のIBがある）なら、ステップS211に進み、そのリンクタグより次の拡張IB番号を得、次にステップS212で、いまのIBのインデックスタグに\$FFFFを記録して未使用IBとし、ステップS213において、そのIBが属するページバッファを指すページタグの更新フラグをセットし、ステップS214において、そのインデックスページのページアカウンをインクリメントする。

【0222】次にステップS215において、拡張IB番号を128で割り、インデックスページ番号を得る。そしてステップS216で、そのインデックスページのデータを、ページバッファに読み込み（図17に示した場合と同様の処理を行う）、次にステップS217で、

変数 i を0で初期化する。

【0223】次にステップS218で、第 i (いまの場合、第0) のエクステント記述子を得て、割当ブロック数と開始割当ブロック番号を得る。次にステップS219で、アロケーションマップにおいて、上記開始割当ブロック番号のビット位置から、割当ブロック数分のビットを0にして、データ領域を解放する。

【0224】次にステップS220において、ファイルサイズからエクステントサイズ、即ち、割当ブロック数×割当ブロックサイズを引くとともに、変数 i をインクリメントする。

【0225】次にステップS221において、ファイルサイズを判定し、0または負ならば、ステップS224で、その IB のインデックスタグに \$FFFF を記録して未使用 IB とし、ステップS225において、その IB が属するページバッファを指すページタグの更新フラグをセットし、ステップS226において、そのインデックスページのページアカウンをインクリメントし、終了する。

【0226】ステップS221で、ファイルサイズが正ならば、ステップS222で、 $i = 4$ でないことを確認し、次にステップS223で第 i (いまの場合、第1) のエクステント記述子の最上位ビット (リンクフラグ) を判定し、リンクフラグが0ならば、ステップS218に戻って、次のエクステントを解放し、リンクフラグが1ならば、ステップS211に戻って、次の拡張 IB を得る。

【0227】ステップS222において、 $i = 4$ のとき、ファイルサイズが正であるのに次の拡張 IB がないことを意味するので、エラーであり、エラー処理を行う。

【0228】〈実施例4〉実施例1の説明におけるインデックスタグの説明 (図9) において、ディレクトリ属性の IB のファイルサイズを0としたが、ディレクトリファイルに実体のデータを持たせることもできる。このようにした場合、より高度なファイル管理方法を実現することができる。図29は、その実施例を表している。

【0229】即ち、この実施例においては、 $IB = 3$ 、 $IB = 4$ 、 $IB = 5$ が、いずれも、最初の拡張フラグが1とされ、基本型 IB とされている。 $IB = 3$ のインデックスタグは0とされ、ルートディレクトリ直下のファイルであることが示されている。また、属性コードは1とされ、ディレクトリ属性のファイルとされている。ファイルサイズは4バイトとされ、割当ブロック24から、1割当ブロックの領域が割り当てられている。 $IB = 4$ のインデックスタグは3とされ、 $IB = 3$ のディレクトリ直下のファイルとされている。また、属性コードは0とされ、データ属性のファイルとされている。さらに、ファイルサイズは212バイトとされ、割当ブロック25から、1割当ブロックの領域が割り当てられてい

る。 $IB = 5$ のインデックスタグは3とされ、 $IB = 3$ のディレクトリ直下のファイルとされている。また、属性コードは0とされ、データ属性のファイルとされている。さらに、ファイルサイズは180バイトとされ、割当ブロック26から、1割当ブロックの領域が割り当てられている。

【0230】図29に示すように、この実施例の場合、 $IB = 3$ の割当ブロック24は、各2バイトからなる IB 番号のリストで構成され、そこに子ファイルの IB 番号が記録される。即ち、この実施例では、 $IB = 4$ と $IB = 5$ の2つの IB 番号が記録されている。

【0231】前述のインデックスタグのみによるファイルの階層的管理方法においては、新しいファイルの記録や、 IB 番号がわかっているファイルの再生は、高速に行うことが可能であるが、同一のディレクトリに属する子ファイルを検索する場合、インデックステーブルをすべて走査する必要があるため、時間がかかることになる。しかしながら、この実施例のようにすれば、ディレクトリファイルから子ファイルを検索することができ、子ファイルの検索が高速化される。

【0232】〈実施例5〉図30は、他の実施例を表している。図29の実施例と異なり、この実施例においては、 $IB = 3$ のファイルサイズが32バイトとされ、割当ブロック24のエントリは、16バイトとされ、そこには IB 番号 (2バイト) だけでなく、ファイル名 (14バイト) も記録されるようになされている。

【0233】図30の実施例においては、 $IB = 3$ のディレクトリの下に、割当ブロック24に、"CONFIG. SYS" および "AUTOEXEC. BAT" という2つのファイルが記録されている。勿論、 $IB = 3$ のディレクトリファイル自身にも、ファイル名を付けることができ、そのファイル名は、 $IB = 0$ のルートディレクトリファイルに記録される。

【0234】上記の構成により、各 IB に登録されたファイルは、ファイル名を持つことができ、コンピュータのOSのように、ファイル名を必要とする、従来のファイルシステムにも容易に適用することができる。

【0235】尚、図30においては、簡単のため、ディレクトリファイルの構成を、 IB 番号と固定長のファイル名のみとしたが、次のようにすることも可能である。

【0236】(1) ファイル名の長さ、各エントリの長さを記録し、可変長にする。

(2) ファイル名のみならず、記録者名や著作権表示等を記録したり、ウインドウシステムで使用するアイコンデータを記録するなど、ファイル固有の諸情報を、合わせて記録する。

(3) ファイル名をキーとして、2分木、Bツリー、ハッシュ表等の構成とし、子ファイルが多数の場合のファイル名検索を高速化する。

(4) 各エントリ中に、再生順序の番号を記録するな

ど、子ファイルの順番を管理する手段を設ける。

【0237】〈実施例6〉前記実施例4と実施例5の説明において、属性コードは、0をデータ属性、1をディレクトリ属性としたが、属性コードを複数ビットにより構成し、前記ディレクトリファイルの構成方法を、用途に応じて複数種類持つようにすることができる。図31は、その実施例を表している。

【0238】図31において、IB=0のルートディレクトリの子ファイルとして、IB=1、IB=2、IB=3と、3つのIBが示されている。

【0239】IB=1の属性コードは0とされ、データ属性のファイルとされている。そして、そのデータは、割当ブロック10に記録されている。IB=2の属性コードは1とされ、第1の種類のディレクトリファイルとされている。図31においては、第1の種類のディレクトリファイルとして、前記実施例4に準じ、IB=2で指定される割当ブロック11に、子ファイルのIB番号リストが記録されている。IB=3の属性コードは2とされ、第2の種類のディレクトリファイルとされている。この図においては、第2の種類のディレクトリファイルとして、前記実施例5に準じ、IB=3で指定される割当ブロック12に、子ファイルのIB番号とファイル名のリストが記録されている。

【0240】同様にして、用途に応じた多数の種類のディレクトリファイルを、同一ファイルシステム上に混在させることができる。このように、ディレクトリファイルの種類が多くなると、応用機器によっては、すべての種類をサポートできない場合があると思われる。しかし、前記実施例1において説明したように、ファイルの階層構造に関する情報は、IB中のインデックスタグにも記録されているため、そのような場合においても、ファイルの階層構造を簡単に知ることができる。

【0241】尚、本発明はこの実施例にのみ限定されるものではなく、例えば、次のような変形が可能である。

【0242】(1) 割当ブロックの管理をアロケーションマップ(ビットマップ)ではなく、FATで行なう。

【0243】この場合、エクステント記述子を、割当ブロック数と開始割当ブロック番号ではなく、FATの開始クラスタで表すことができる。

【0244】(2) 割当ブロックの管理をアロケーションマップ(ビットマップ)ではなく、例えば、本出願人が特願平5-114863号の実施例3として先に提案したような、空きの割当ブロックの番号を直接記録したノードマップまたはノードリストで行なう。

【0245】この場合、エクステント記述子を、割当ブロック数と開始割当ブロック番号ではなく、ノード番号で表すことができる。

【0246】(3) IBのサイズを16バイトではなく、他のサイズにし、基本型IBにおける要素の一部(属性コード、タイムスタンプ)を省略するか、または

他の要素(ファイル名、ユーザーIDなど)を記録する。

【0247】

【発明の効果】以上説明したように、本発明によれば、ファイルの階層的管理に必要な情報を、全てインデックス領域に集中したことにより、文字入力手段を持たない機器においても、従来に比較して、小さなメモリ容量で、高速なファイルアクセスが実現できる。

【0248】また、上記インデックステーブルをページ単位で管理できる機構にすることにより、より少量のメモリでより高速なファイルアクセスが実現できる。

【図面の簡単な説明】

【図1】本発明のファイル管理方法を適用したデータ記録媒体のフォーマットを説明する図である。

【図2】図1の実施例をさらに具体化したフォーマットを説明する図である。

【図3】インデックスブロックの構成を説明する図である。

【図4】本発明の実施例1におけるデータ記録処理を説明するフローチャートである。

【図5】図4に続くフローチャートである。

【図6】実施例1における再生処理を説明するフローチャートである。

【図7】実施例1における消去処理を説明するフローチャートである。

【図8】グループタグによるインデックステーブルを説明する図である。

【図9】インデックスタグによるインデックステーブルを説明する図である。

【図10】インデックスタグによる階層構造を説明する図である。

【図11】インデックスブロックの基本型の構成を説明する図である。

【図12】インデックスブロックの拡張型の構成を説明する図である。

【図13】エクステント記述子の構成を説明する図である。

【図14】インデックスブロックの拡張を説明する図である。

【図15】インデックスブロックのキャッシュシステムを説明する図である。

【図16】ページタグテーブルを説明する図である。

【図17】ページバッファへの読み出し処理を説明するフローチャートである。

【図18】ページタグテーブルの動作を説明する図である。

【図19】ページタグテーブルの他の動作を説明する図である。

【図20】ページバッファの書き込み処理を説明する図である。

【図21】 ページアカウントテーブルの具体例を説明する図である。

【図22】 インデックスブロックのキャッシュシステムにおける記録処理を説明するフローチャートである。

【図23】 図22に続くフローチャートである。

【図24】 インデックスブロックのキャッシュシステムにおける再生処理を説明するフローチャートである。

【図25】 図24に続くフローチャートである。

【図26】 インデックスブロックのキャッシュシステムにおける消去処理を説明するフローチャートである。

【図27】 図26に続くフローチャートである。

【図28】 図27に続くフローチャートである。

【図29】 実施例4の構成を説明する図である。

【図30】 実施例5の構成を説明する図である。

【図31】 実施例6の構成を説明する図である。

【図32】 従来のファイル管理方法を適用したデータ記録再生装置の構成例を示すブロック図である。

【図33】 MS-DOSファイルシステムを説明する図である。

【図34】 MS-DOSのルートディレクトリを説明する図である。

【図35】 MS-DOSの階層的ファイル管理を説明する図である。

【図36】 MS-DOSにおける記録処理を説明するフローチャートである。

【図37】 MS-DOSにおける再生処理を説明するフローチャートである。

【図38】 MS-DOSにおける消去処理を説明するフローチャートである。

【図39】 UNIXファイルシステムの構成例を示す図である。

【図40】 iノードリストを説明する図である。

【図41】 UNIXにおける階層的ファイル管理を説明する図である。

【図42】 iノードにおける記録処理を説明するフローチャートである。

【図43】 図42に続くフローチャートである。

【図44】 iノードにおける再生処理を説明するフローチャートである。

【図45】 iノードにおける消去処理を説明するフローチャートである。

【図46】 図45に続くフローチャートである。

【図47】 ミニディスクの構成を説明する図である。

【図48】 ミニディスクにおけるU T O Cセクタ0を説明する図である。

【図49】 ミニディスクにおける記録処理を説明するフローチャートである。

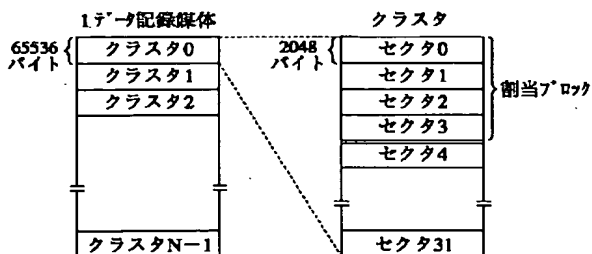
【図50】 ミニディスクにおける再生処理を説明するフローチャートである。

【図51】 ミニディスクにおける消去処理を説明するフローチャートである。

【符号の説明】

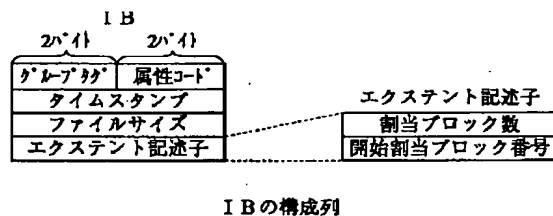
- 1 データ記録媒体
- 2 媒体駆動部
- 3 メモリ
- 4 CPU
- 5 データ入力部
- 6 データ出力部

【図1】



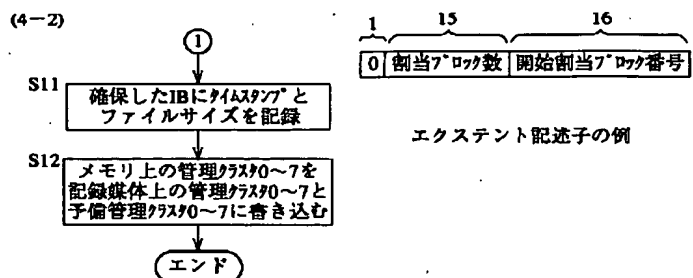
データ記録媒体の物理構成

【図3】

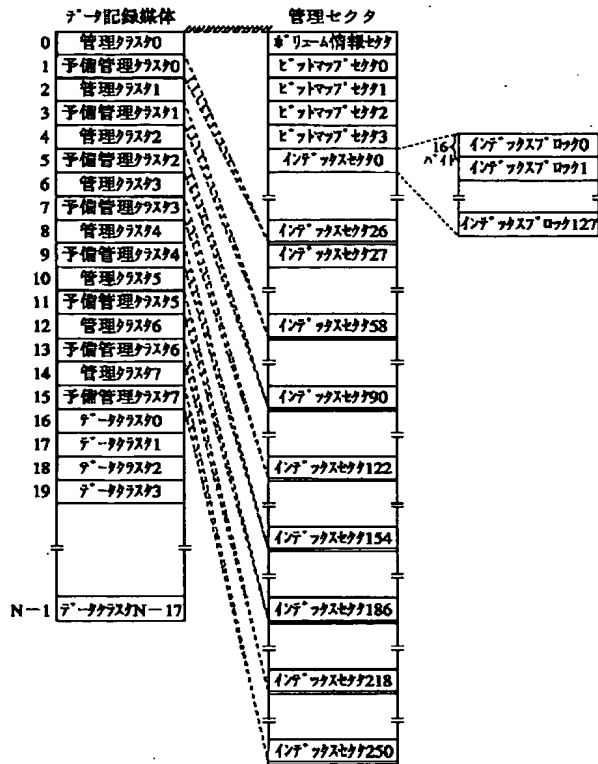


【図5】

【図13】

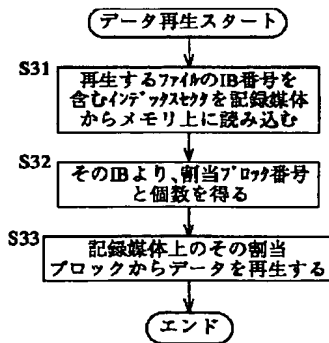


【図2】

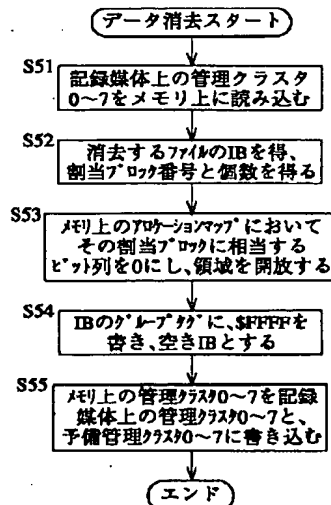


データ記録媒体の論理構成

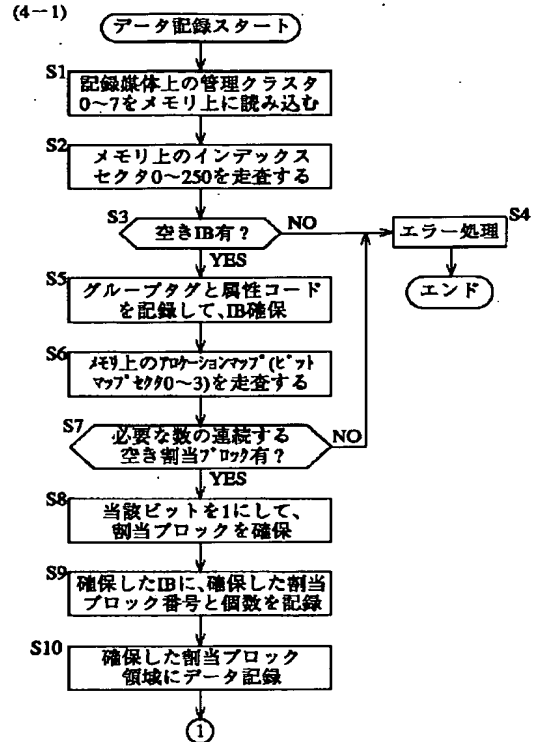
【図6】



【図7】



【図4】



【図8】

IB番号	インデックステーブル	
0	0	
	65536	
	8	0
1	0	
	131072	
	16	8
2	1	
	8192	
	1	24
3	2	
	131072	
	16	25
4	1	
	1048576	
	128	41
5	1	
	8192	
	170	169
6	\$FFFF	

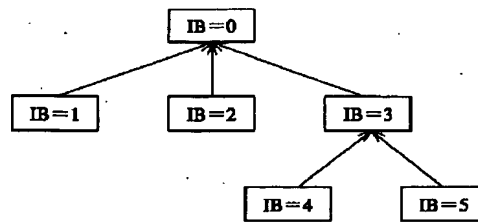
インデックステーブル(グループタグ)

【図9】

IB番号	インデックステーブル
0	0 1
	0
	\$FFFF 0
1	0 0
	131072
	16 0
2	0 0
	65536
	8 16
3	0 1
	0
	\$FFFF
4	3 0
	8192
	1 24
5	3 0
	8192
	1 25
6	\$FFFF

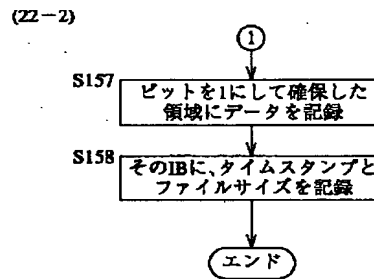
インデックステーブル(インデックスタグ)

【図10】

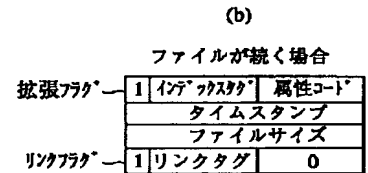
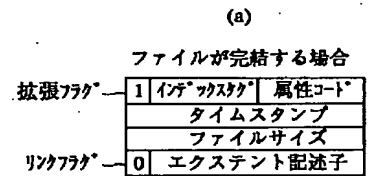


インデックスタグによる階層構造

【図23】

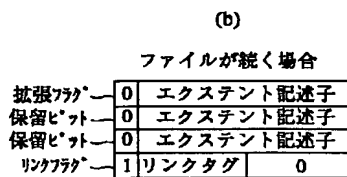
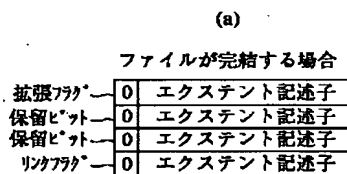


【図11】



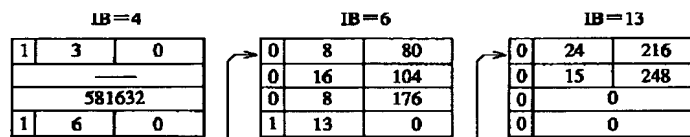
IBの構成列(基本型)

【図12】

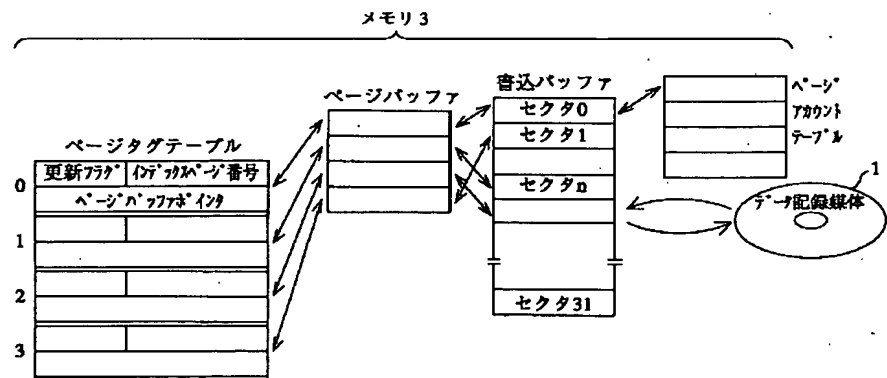


IBの構成列(拡張型)

【図14】

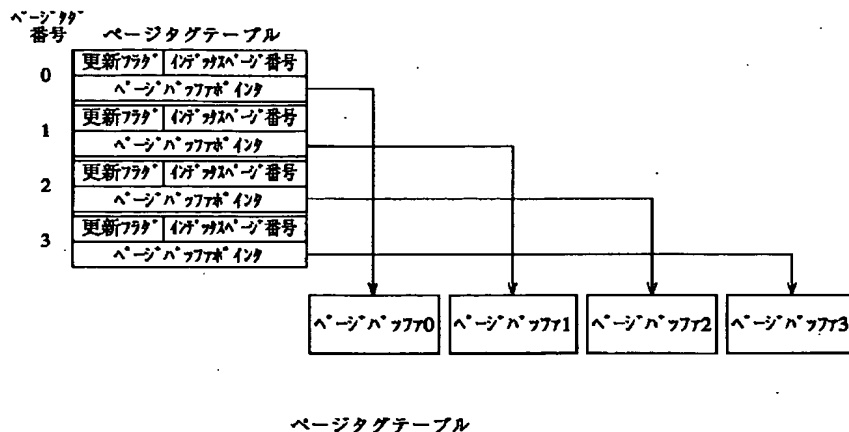


【図15】

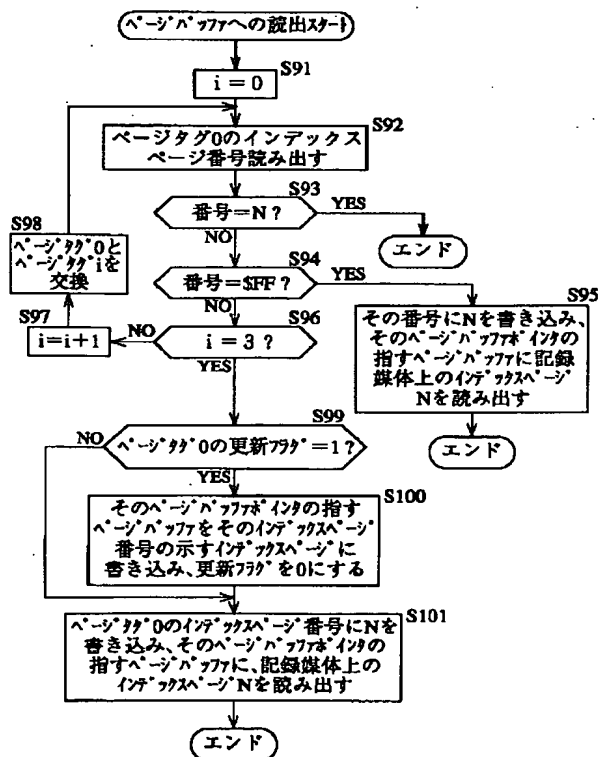


IBキャッシュシステム

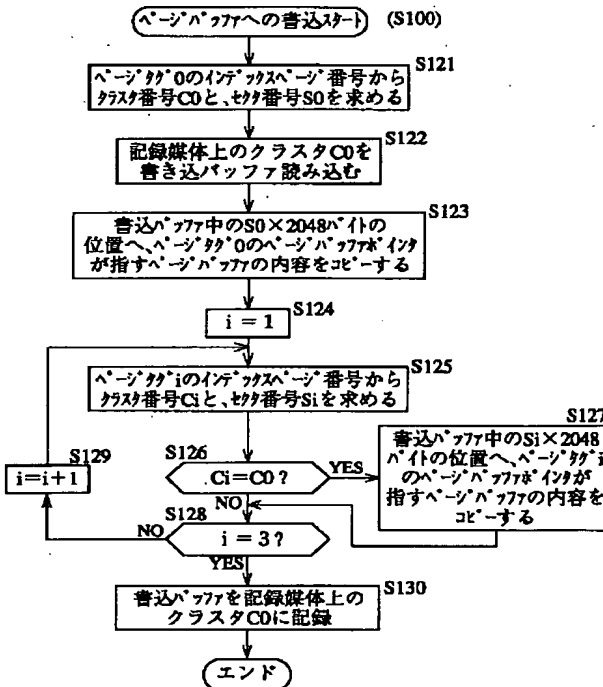
【図 16】



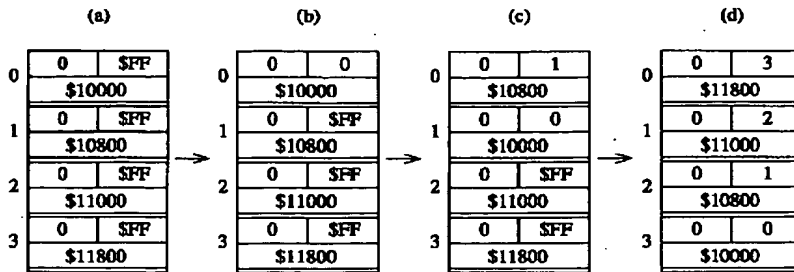
【図 17】



【図 20】

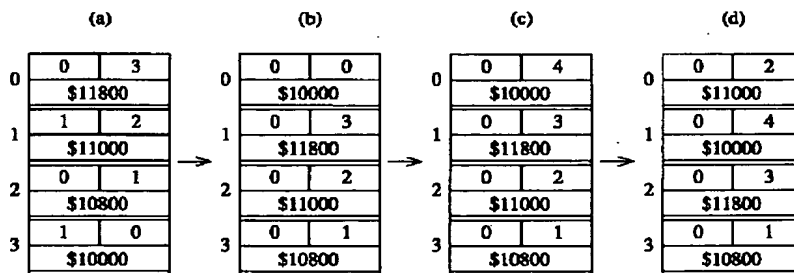


【図 2 2】



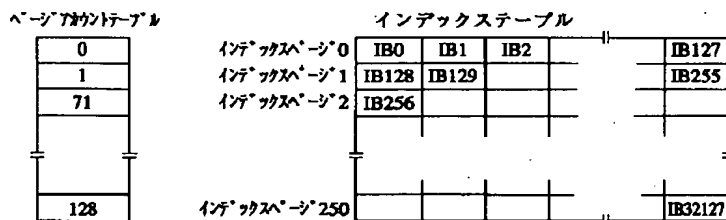
ページタグテーブルの動作1

【図 19】



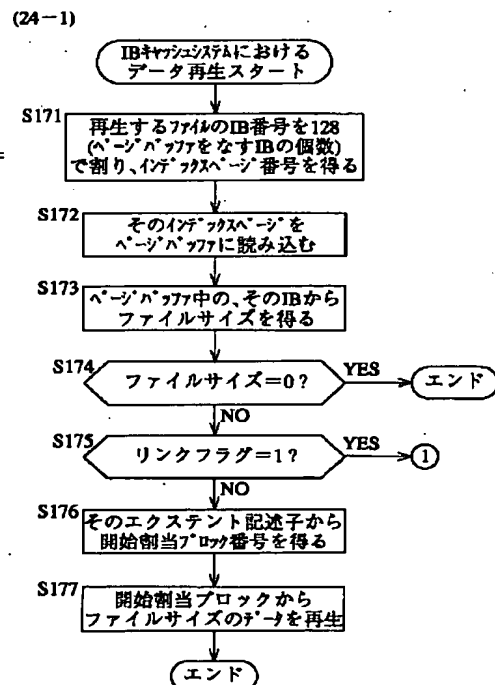
ページタグテーブルの動作2

【図 2 1】

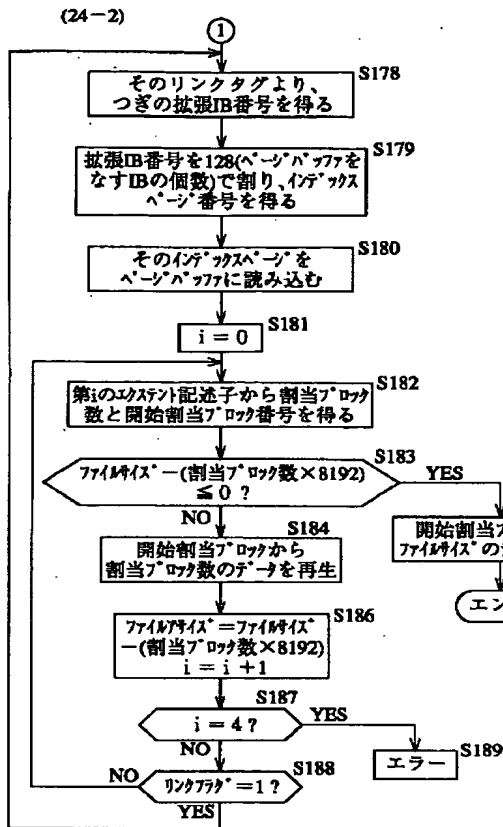


ページアカウントテーブル

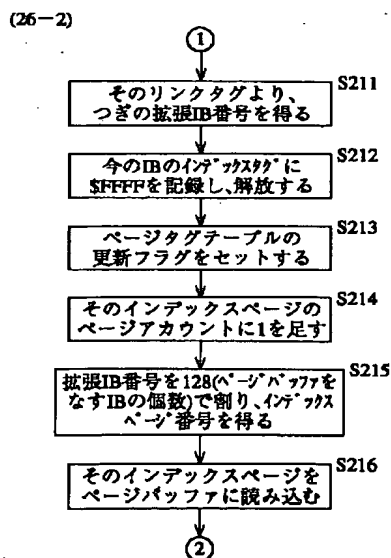
【图 2 4】



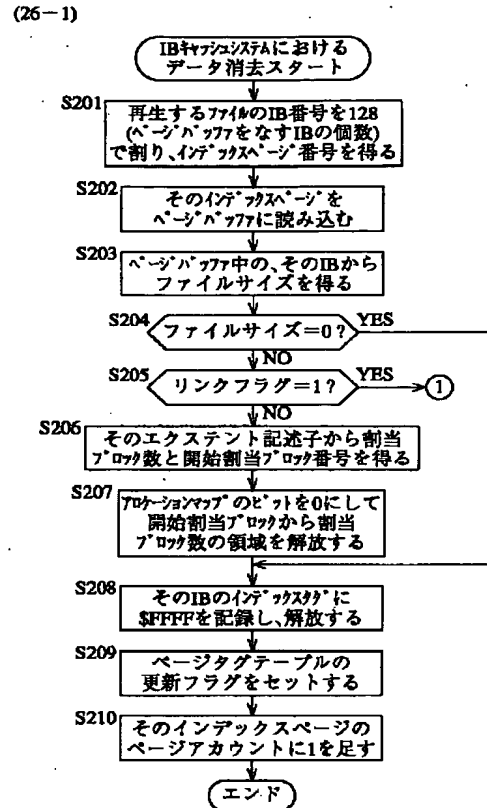
【図25】



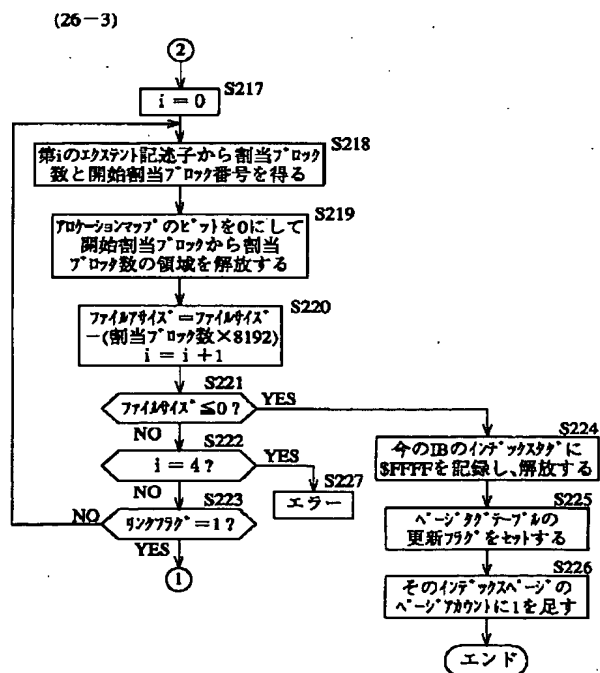
【図27】



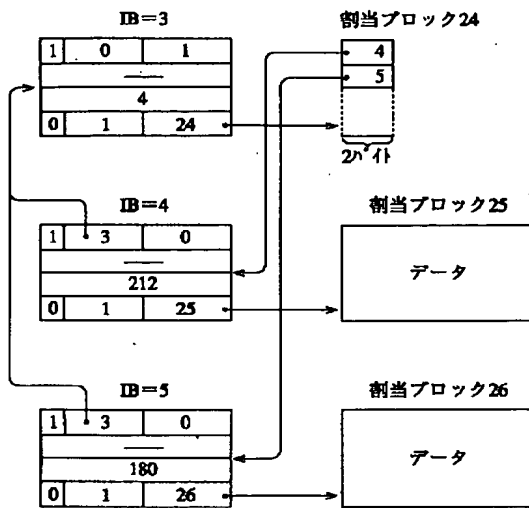
【図26】



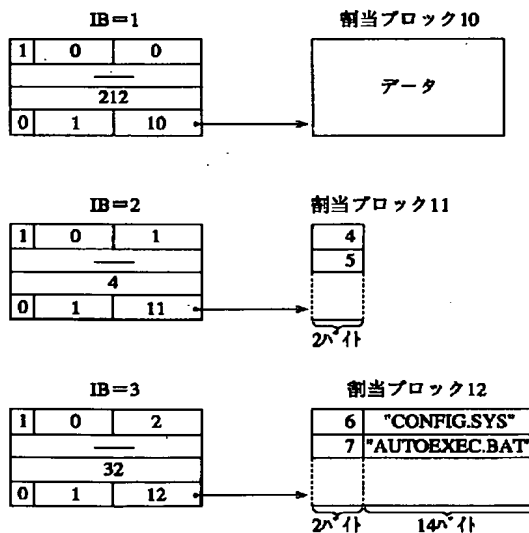
【図28】



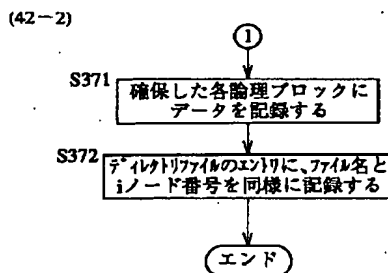
【図29】



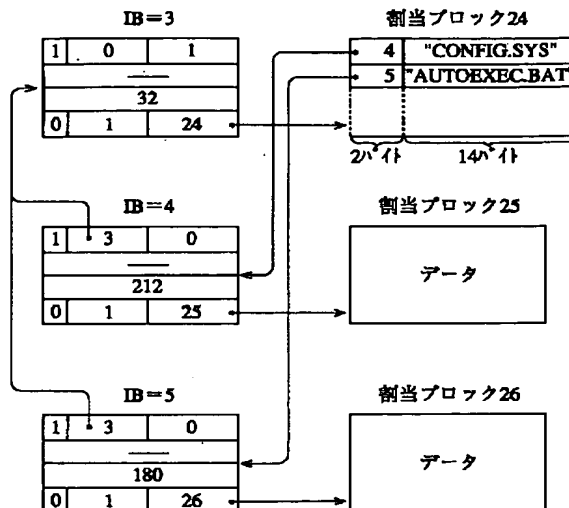
【図31】



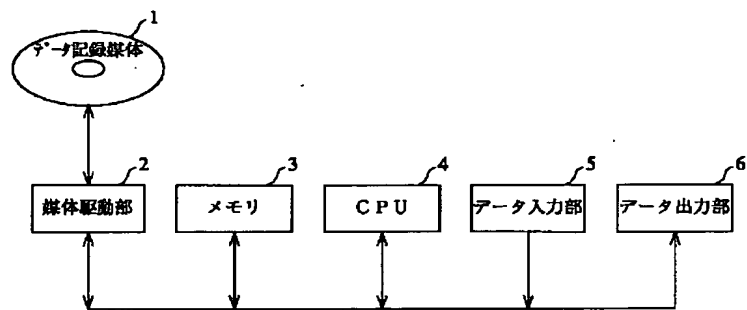
【図43】



【図30】

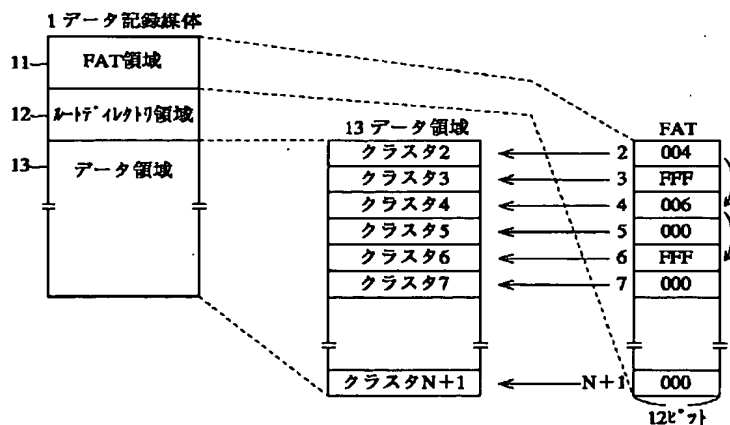


【図32】



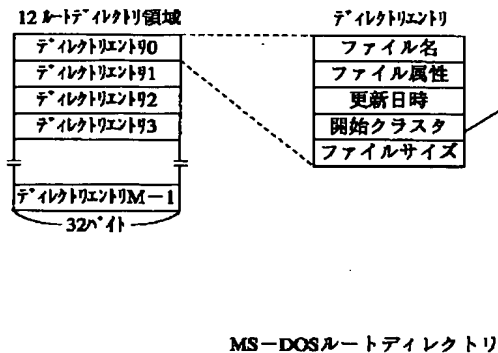
データ記録再生装置

【図33】

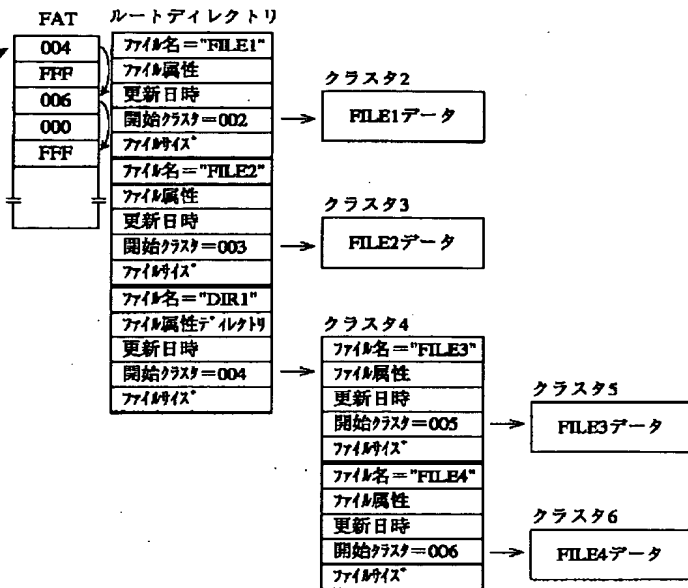


MS-DOSファイルシステム

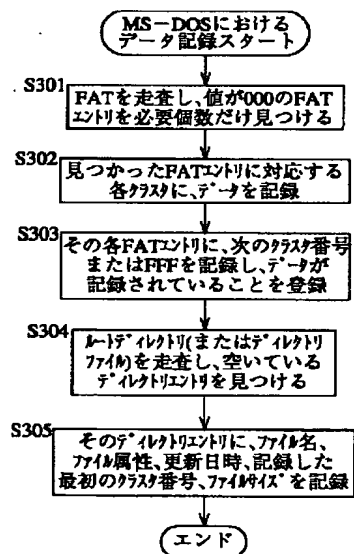
【図34】



【図35】

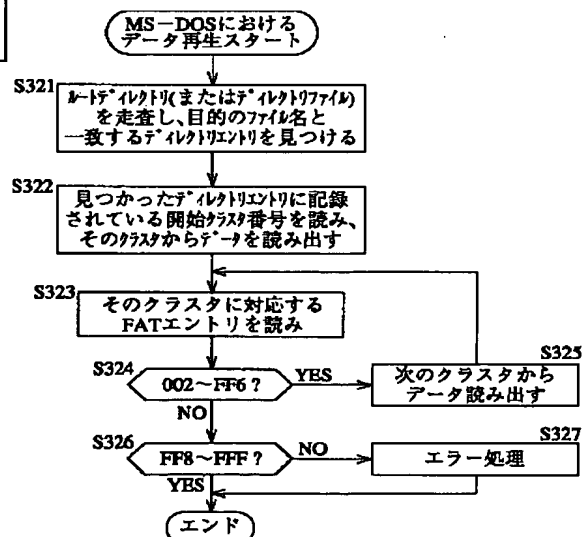


【図36】

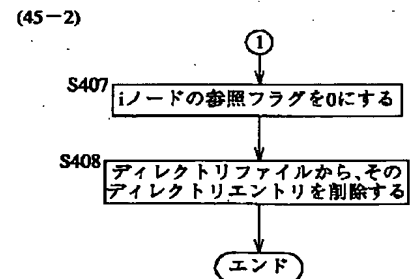


MS-DOS階層的ファイル管理

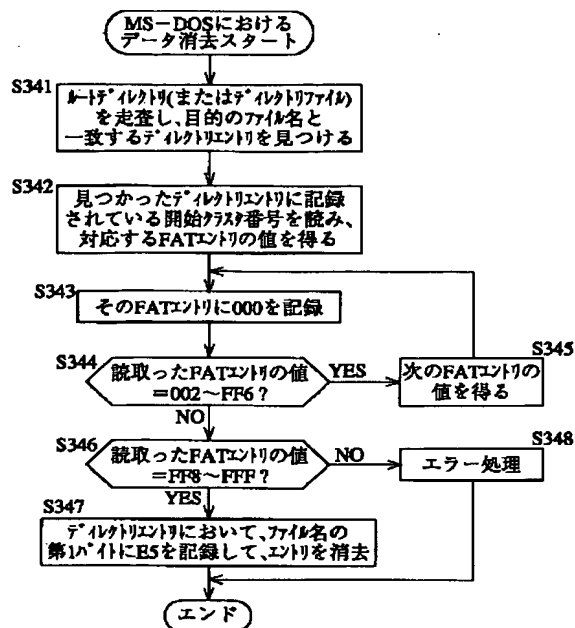
【図37】



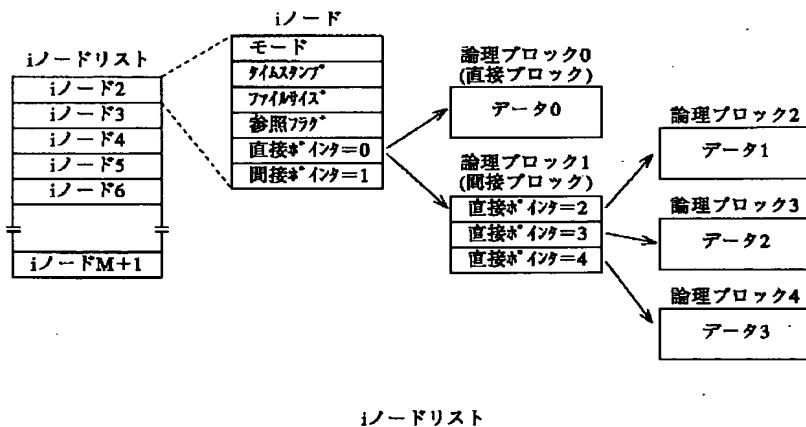
【図46】



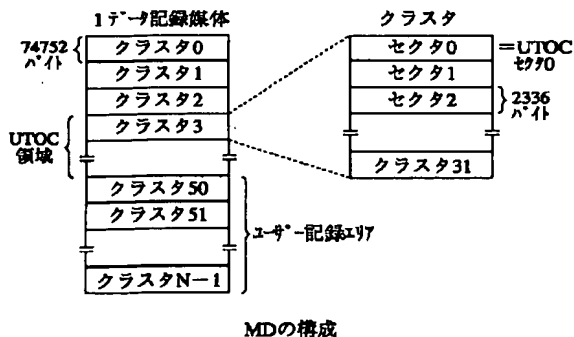
【図38】



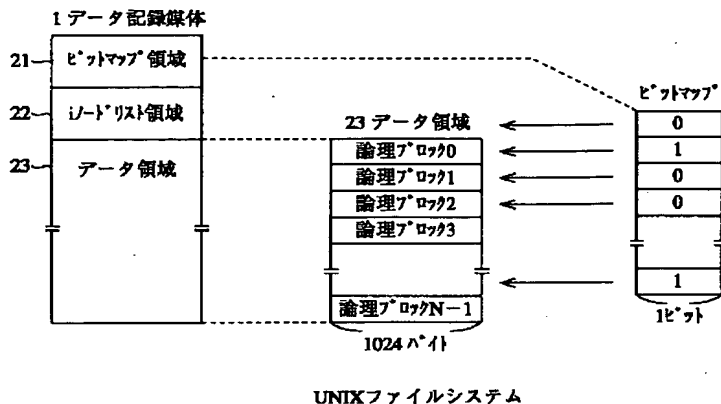
【図40】



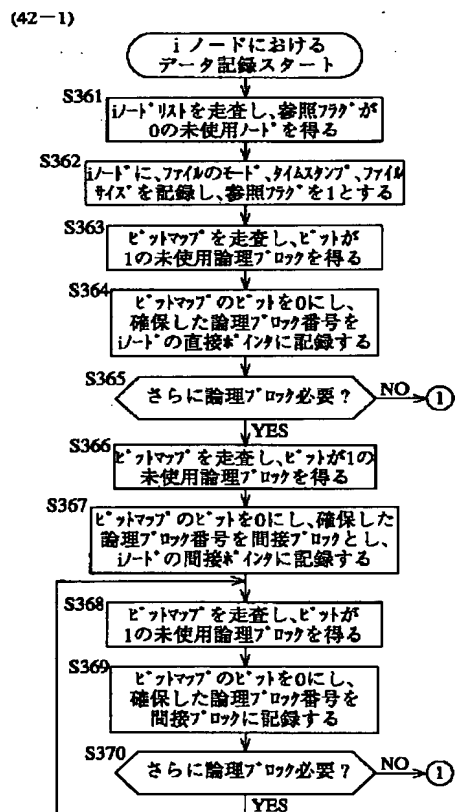
【図47】



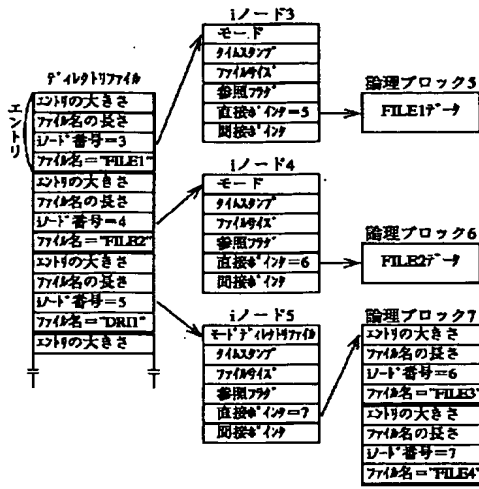
【図39】



【図42】

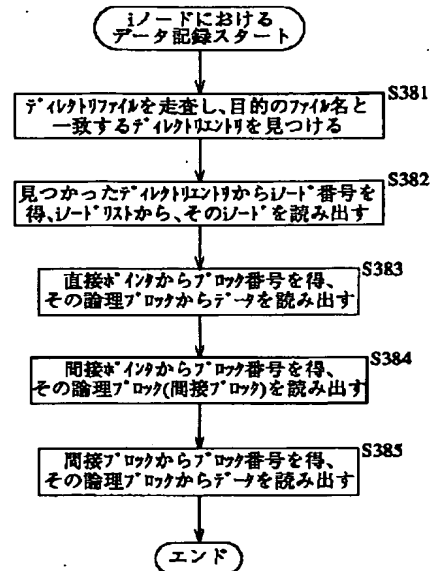


【図41】



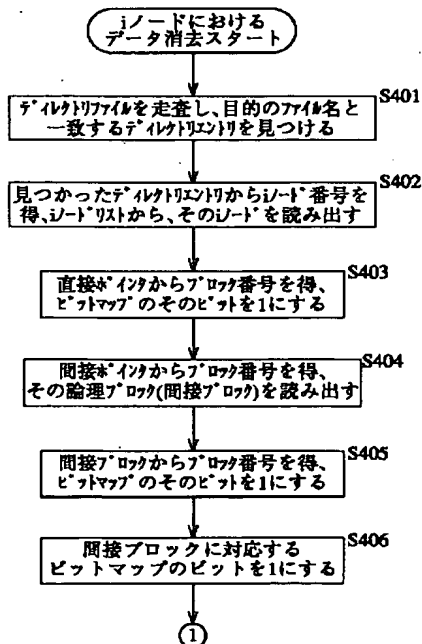
UNIX階層的ファイル管理

【図44】

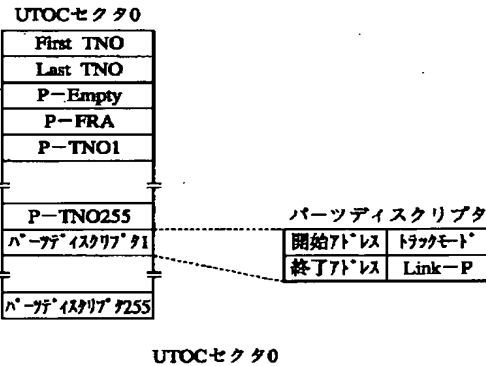


【図45】

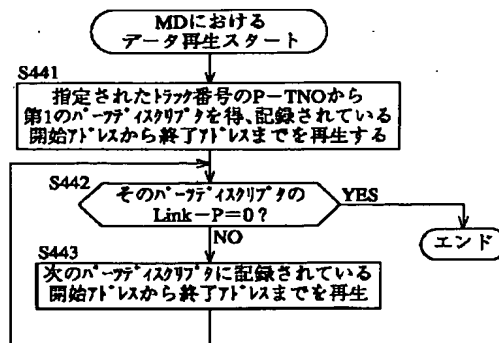
(45-1)



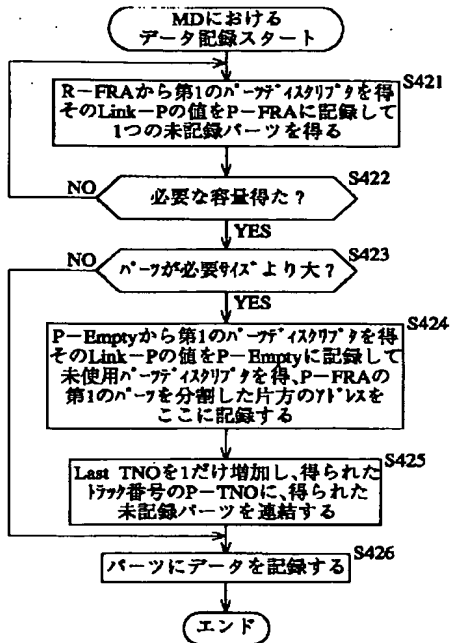
【図48】



【図50】



【図49】



【図51】

